A TRIDENT SCHOLAR PROJECT REPORT

NO. 512

Predicting Naval Academy Performance Using Holistic Personality Analysis in Multidimensional Space

by

Midshipman 1/C Philip B. Smith, USN

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
7/12/21		

4. TITLE AND SUBTITLE	**5a. CONTRACT NUMBER**
Predicting Naval Academy Performance Using Holistic Personality Analysis in Multidimensional Space	**5b. GRANT NUMBER**
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	**5d. PROJECT NUMBER**
Smith, Philip, B.	**5e. TASK NUMBER**
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	**8. PERFORMING ORGANIZATION REPORT NUMBER**

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
U.S. Naval Academy Annapolis, MD 21402	**11. SPONSOR/MONITOR'S REPORT NUMBER(S)** Trident Scholar Report no. 512 (2021)

12. DISTRIBUTION / AVAILABILITY STATEMENT

This document has been approved for public release; its distribution is UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Psychological models of personality have used trait measures to index individuals with respect to specific traits or categorical types to bin individuals into defined personality types. Both of these approaches may be suboptimal for predicting performance. Categorical models rely on rough dichotomization of data and trait models may overfit variance to rigid traits and obscure trait interaction effects. This project compared predictions of midshipmen performance and outcomes at the United States Naval Academy, specifically comparing predictions derived from regression techniques with standard traits and type variables with predictions using machine learning techniques, such as k-nearest neighbors and boosted random forests. Using data from recent Naval Academy graduates (N = 725), we first applied traditional penalized regression techniques to predict performance, specifically academic and military order of merit at graduation, using standard personality traits, types, and values. These predictions served as the baseline for assessing the quality of prediction from the selected machine learning techniques. We next built, optimized, and analyzed machine learning models, finding that their accuracies were, at best, at the level of traditional penalized regression models. Finally, we examined the optimization process of the machine learning models to identify potential optimum dimensionalities for personality predictions, finding it matches currently accepted models of personality. While the machine learning models were more complex, computationally expensive, and less interpretable, we found they did not outperform regression models.

15. SUBJECT TERMS

Personality, performance, facets, machine learning

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		31	**19b. TELEPHONE NUMBER** *(include area code)*

U.S.N.A. --- Trident Scholar project report; no. 512 (2021)

PREDICTING NAVAL ACADEMY PERFORMANCE USING HOLISTIC PERSONALITY ANALYSIS IN MULTIDIMENSIONAL SPACE

by

Midshipman 1/C Philip B. Smith
United States Naval Academy
Annapolis, Maryland

Certification of Advisers Approval

Associate Professor Kevin M. Mullaney, CAPT, USN
Department of Leadership, Ethics, and Law

Professor William N. Traves
Mathematics Department

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

USNA-1531-2

Abstract: Psychological models of personality have used trait measures to index individuals with respect to specific traits or categorical types to bin individuals into defined personality types. Both of these approaches may be suboptimal for predicting performance. Categorical models rely on rough dichotomization of data and trait models may overfit variance to rigid traits and obscure trait interaction effects. This project compared predictions of midshipmen performance and outcomes at the United States Naval Academy, specifically comparing predictions derived from regression techniques with standard traits and type variables with predictions using machine learning techniques, such as k-nearest neighbors and boosted random forests. Using data from recent Naval Academy graduates ($N = 725$), we first applied traditional penalized regression techniques to predict performance, specifically academic and military order of merit at graduation, using standard personality traits, types, and values. These predictions served as the baseline for assessing the quality of prediction from the selected machine learning techniques. We next built, optimized, and analyzed machine learning models, finding that their accuracies were, at best, at the level of traditional penalized regression models. Finally, we examined the optimization process of the machine learning models to identify potential optimum dimensionalities for personality predictions, finding it matches currently accepted models of personality. While the machine learning models were more complex, computationally expensive, and less interpretable, we found they did not outperform regression models.

Keywords: personality, performance, facets, machine learning

Acknowledgements: We owe special thanks to Dr. Celeste R. Luning, Class of '67 Leadership Research Fellow; CDR David M. Wallace, USN, Ph.D., Permanent Military Professor in the Leadership, Ethics, and Law Department; the Office of Institutional Research, Planning, and Analysis at USNA and the USNA Human Research Protection Program Office.

Table of Contents

Performance and Personality

Once upon a time, the philosophy of the United States Naval Academy was to serve as a filter to select and commission only the best Midshipmen. As such, attrition rates were as high as 39% in 1975 (Staats, 1976). The current philosophy of the Naval Academy is to filter for the best qualified candidates through the admissions process and actively seek to commission all admitted students. To achieve this end, it is useful to be able to predict how midshipmen will perform in order to allocate resources to help each successfully graduate and commission. The goal of the current study is to investigate the utility of more recent machine learning analysis techniques in predicting performance as compared to more traditional methodologies. This study applies these techniques to outcomes relevant to the Naval Academy and its commissioning processes and uses psychological constructs and instruments already being used at the Naval Academy as part of the leadership development enterprise.

A primary predictor of performance in many high stakes settings is personality. Oswald and Hough (2011) provide a useful review of the history and current techniques of personality assessment for selection purposes. Although the current research is not intended to generate selection tools, considering the use of personality in selection is useful in that selection is primarily about predicting performance.

The Big Five personality traits (Openness, Consciousness, Extraversion, Agreeableness, and Neuroticism) and their thirty facets are measured with a variety of self-report personality tests. One of the more accessible tests is the IPIP-NEO-120 (Johnson, 2014), a shorter version of the original 300-item test (Goldberg, 1999). The IPIP-NEO-120 test is an online, self-report test measuring 30 personality facets which aggregate to the Big Five traits. Importantly, the Big Five model is based on a factor analysis of the English language, not in cognitive psychology, but is still widely considered the most scientific model of personality. Studies have shown the Big Five can predictive of performance in several different situations. For instance, Lado and Alonso (2017) found that consciousness and neuroticism were strong predictors of overall job performance in several different careers. Research using the lower-level facets of the Big Five has been proven to be more powerful than just the overarching traits (Oswald & Hough, 2011; Mottus & Rozgonjuk, 2021). As such, it may be useful to include even more low-level variables, in the form of other psychological tests.

Our second test is the Values in Action-Inventory of Strengths (VIA). Developed as a "manual of the sanities," VIA is one of the major products of the field of positive psychology, which posits "that character strengths are the bedrock of the human condition and that strength-congruent activity represents an important route to the psychological good life" (Peterson and Seligman, 2004). As a whole, positive psychology focuses on increasing overall wellbeing by informing individuals their strengths and how to capitalize on them. Although Christopher Peterson and Martin Seligman constructed the test primarily to assist the individuals taking it, the VIA has been proven to be predictive in several situations. For instance, Kern and Bowling

(2015) found significant negative correlations between factor analyzed VIA scores and law school admissions tests among incoming law school students. In other instances, individual values can be proven to have a significant impact on lives. For instance, another study found perseverance, love of learning, humor, fairness, and kindness associated with better grades for college students (Lounsbury, Fisher, Levy, & Welsh, 2009).

Our final test administered is the Myers-Briggs Type Indicator. A popular model of personality, the MBTI is an archetype of type methodology in that it attempts group each individual into 16 different personality types using four dichotomous measurements (Introversion/Extraversion: whether a person draws energy from the outer world versus their own ideas and impressions; Sensing/Intuiting: whether a person prefers to use senses to gather data or to rely on relationships and patterns; Thinking/Feeling: whether a person prefers objective logic or personal values; and Judging/Perceiving: whether a person prefers organization or flexibility). An underlying assumption of the MBTI is that the different dimensions are interactive. For example, an extraverted and sensing individual will express their extraversion in a different manner than an extraverted and intuitive individual (Myers, McCaulley, Quenk, & Hammer, 1998). Unlike the previous two tests, which can only categorize someone as having "more" or "less" of a specific personality trait or value, the MBTI assigns individuals numbers ranging from 0 to 30 which reflect the degree of confidence that the model correctly classified a respondent into a particular type category. Although it is considered less frequently than the Big Five and the VIA, the MBTI still is useful in predictive studies. For instance, Kim, Park, Seo, and Ihm (2014) found that judging was a notable predictor of problem-based learning success in graduate students.

While it may seem odd at first glance to include three different personality tests based on different methodologies in one study, we believe that it is one of this study's strengths. Over the past several years, an increase in focus on predictive power rather than a focus on in-depth explanations of the causes of certain behaviors has allowed a broader view of personality, especially when coupled with developing machine learning tools and techniques (Yarkoni & Westfall, 2017). Together, we believe the three personality tests may be able to yield more accurate results and a deeper understanding of performance.

Current Study

To form our predictions for midshipman performance, we used previously gathered data from the Brigade of Midshipman at the United States Naval Academy. Midshipmen receive up to three different personality tests during their four years at the Naval Academy. These are the International Personality Item Pool (IPIP-120), which measures the Big Five domains and their 30 facets; the Values in Action Inventory of Strengths (VIA), which measures 24 character strengths; and the Myers-Briggs Type Indicator (MBTI-M), which categorizes subjects into 16 personality types based on four measurements.

Tests are administered during core leadership classes taken by all midshipmen. As some instructors decide not to administer certain tests, not every midshipman takes all three personality tests. We limited our study to midshipmen who took every test, allowing us to take all personality factors into consideration.

Midshipmen are measured on many different performance metrics throughout their time at USNA, but arguably the two best measures of overall performance are Academic Order of Merit (AOM) and Military Order of Merit (MOM). Both of these measures are calculated by ranking each midshipman against all peers in his or her class on cumulative academic or military performance, respectively. AOM is based entirely off of grades, while MOM is impacted by physical performance, adherence to honor and conduct systems, peer and superior rankings, and more. Viewed together, AOM and MOM seem to represent total midshipman performance at USNA.

Methods

Since the data used for this study was collected from midshipmen at USNA, we applied for and received permission from the Human Research Protection Program to gather and study the data and publish this analysis.

Main Analysis

Participants. From the 2382 midshipmen in the classes of 2017 and 2018, 725 took all three personality tests. We can assume that the selection of these midshipmen was random, since midshipmen are evenly distributed among these core classes and do not know which instructors administer which personality tests when choosing a schedule. Of these 725 midshipmen, 189 were female and the remaining 536 were male. This gender ratio is approximately representative of the Brigade of Midshipmen as a whole. Age was unavailable, but every midshipman must be between the ages of 21 and 26 upon graduating.

Measures. To measure the Big Five and its facets, participants completed the IPIP-120 (Johnson, 2014), a 120-item self-report personality questionnaire based on the 300-item International Personality Item Pool (Goldberg, 1999). We did not have access to item-level data, but were able to view information for all 30 facets and the combination of those facets into the overarching five traits. This test was administered during the fifth or sixth semester at USNA during a core leadership course, and accounts for 35 of the independent variables.

To measure the Values in Action, participants completed the VIA Inventory of Strengths (Peterson & Seligman, 2004), a 240-item self-report survey on a five-point Likert scale. While

we had the item level data for each question for this measure, we only used the compiled scores for values to match the other two personality tests. This test was administered during the first or second semester at USNA in a core leadership course, and accounts for 24 of the independent variables.

To measure the Myers-Briggs Type Indicator, participants took the Form M MBTI test (Myers, McCaulley, Quenk, & Hammer, 1998), a 93-item self report survey. This gave each person a four-letter personality type (again, extraversion/introversion, sensing/intuition, thinking/feeling, and judging/perceiving), as well as a score from 1 to 30 for each letter. We arbitrarily assigned introversion, intuition, feeling, and perceiving as negatives to be able to construct continuous scales from -30 to 30 for each of the four categories. This test was administered during the first or second semester at USNA in a core leadership course, and accounts for four of the independent variables.

Both of our dependent variables are calculated by the Naval Academy to judge midshipman performance (Carter, 2017). To calculate Academic Order of Merit, midshipmen are ranked within their class on their Academic Quality Point Rating (AQPR). Calculated similarly to a grade point average, AQPR is equal to the sum of the grade received in a course (A=4, B=3, C=2, D=1, F=0) times the number of credits for that course, all divided by the total number of credit hours taken. There are no pluses or minuses associated with grades, and midshipmen must maintain an AQPR above 2.00 to be academically satisfactory. AOM is calculated with a simple ranking of midshipmen, with 1 as the highest AQPR and the largest AOM in the class as the lowest AQPR.

Military Order of Merit is calculated by ranking midshipmen within their class on their Military Quality Point Rating (MQPR). Factors included for consideration include physical education, athletic performance, aptitude (consisting of rankings by peers, upperclass midshipmen, and commissioned officers and senior enlisted), and conduct. The weighting of each of these factors changes as midshipmen continue their time at USNA, and the instruction governing the calculation of MQPR is available on USNA's webpage. MOM is calculated based off of MQPR in the same manner as AOM. While the construction of both AOM and MOM may seem complicated, it is sufficient to consider them as academic and military class standing, respectively.

Analysis. All models were constructed using the *scikit-learn* package (Pedregosa, et al., 2011). Data was collected from the Naval Academy's Office of Institutional Research, Planning, and Analysis and aggregated in Jupyter Notebook (Kluyver, et al., 2016).

After removing midshipmen who had not taken all personality tests, we shuffled midshipman data and divided it into training, development, and testing sets, consisting of 60%, 20%, and 20% of the data, respectively. A preprocessing model was then trained on the training set and applied to all three sets, standardizing all personality metrics onto the range $[0, 1]$ to account for the different measurement ranges of each test.

For our first model, we trained a basic linear regression model for both AOM and MOM on our training set. We computed root mean squared errors for both dependent variables predicted on the development set and analyzed beta weights. These models, however, faced a significant methodological issue, in that many of the eigenvalues were extraordinarily small, reaching as low as 2.6e-27, suggesting we potentially have a large number of intercorrelated independent variables. This was an expected problem—see Appendix A for a correlation matrix of the pre-standardized independent variables.

To overcome this, we implemented an elastic net regression, which is designed to manage large numbers of intercorrelated predictors by shrinking regression coefficients towards zero, leaving only the strongest predictors. Elastic net functions as a combination of the ridge (Hoerl & Kennard, 1970) and Least Absolute Shrinkage and Selection Operator (LASSO) regressions (Tinshirani, 1996). Ridge regression applies a shrinkage penalty to regression coefficients that depends on the sum of the squares of these coefficients; it tends not to increase model parsimony as it rarely shrinks regression coefficients to zero. LASSO regression applies a penalty that depends on the sum of the absolute values of these coefficients and leads to more parsimonious models, but the model may randomly select only one of many correlated predictors and setting coefficients for others to zero, which means that LASSO solutions are not unique, whereas ridge regression tends to shrink coefficients for correlated predictors toward each other (Friedman, Hastie, & Tibshirani, 2010; Waldmann, Meszaros, Gredler, Fuerst, & Solkner, 2013). Elastic net (Zou & Hastie, 2004) combines the ridge and LASSO penalties, mitigating limitations associated with each of them alone. The elastic net yields parsimonious models (because of the LASSO penalty), in which groups of strongly correlated predictors are similarly (because of ridge penalty), all being either included or excluded from the model (Waldmann, Meszaros, Gredler, Fuerst, & Solkner, 2013; Zou & Hastie, 2004).

To build the model, we first found the parameters α and L for each AOM and MOM using 10-fold cross-validation to minimize the cost functional

$$J(\alpha, L) = \frac{1}{2 * n_{samples}} \left|\left|y_{train} - X_{train}\vec{b}\right|\right|_2^2 + \alpha L \left|\left|\vec{b}\right|\right|_1 + 0.5\alpha(1 - L) \left|\left|\vec{b}\right|\right|_2^2$$

across all folds, where $n_{samples}$ is the number of individuals that are currently being trained on, L is the elastic net mixing parameter, α is a constant that multiplies the penalty terms, and $\vec{b}$ is a vector of the beta weights. In this formula, the second term (containing αL) represents the LASSO penalty, and the third term represents the ridge penalty. After examining our beta weights and parameters, we computed learning curves using the optimized parameters, with training size ranging from 5% to 100% of the training set, in 5% increments, and evaluated the performance of the model on the entire training and development sets. Finally, we produced a final prediction on the otherwise untouched testing set.

Our next model was k Nearest Neighbors (KNN). To predict an unknown variable using n independent variables, we plotted all known data points into n-dimensional space, and

predicted a dependent variable value for each unknown point that was equal to the average of the dependent variables of the k nearest neighbors—the closest known data points to the unknown data point using some distance metric, usually Euclidian (Cover & Hart, 1967). The value of k varies from data set to data set, since it needs to be both small enough to give accurate estimates in different areas of the data set and large enough to prevent individual outliers from dominating predictions. Other hyperparameters besides k include different distance metrics and weight functions used in predictions.

KNN can suffer from the curse of dimensionality, since it is dependent on a relatively dense data set (Prestov, 2013). In common language, the curse of dimensionality says that if we use many different measurements to classify individuals, each individual will look very different from the others. KNN is especially vulnerable to this, since it depends on the overall distance between two data points. Adding an extra dimension can significantly hurt the model if it is not relevant, since that extraneous dimension can make two otherwise similar points appear distant (Hinrichs, Novak, & Wozniakowski, 2011). Even adding additional relevant dimensions will eventually significantly hurt a model, since the excessive dimensionality will force models to make comparisons to more and more differing subjects across greater and greater distances (Murphy, 2012).

To overcome the curse of dimensionality, we used the Singular Value Decomposition (SVD) of the data. A given m by n data matrix D can decompose into $D = U\Sigma V^T$where U is a m\times m matrix whose columns are the eigenvectors of DD^T, V is a $n \times n$ matrix whose columns are the eigenvectors of $D^T D$, and Σ is a $m \times n$ diagonal matrix in which the diagonal is the square roots of the eigenvalues of DD^T in descending order. Importantly, for some $0 < z \leq minm, n$, we can truncate the SVD by letting U_z be equal to the first z columns of U, V_z be equal to the first z columns of V, and Σ_z be the diagonal $z \times z$ matrix, in which the diagonal is the first z entries of Σ. The Eckart-Young-Mirsky theorem has proven that $D_z = U_z \Sigma_z V_z^T$ is the closest rank-z matrix to D (Eckart & Young, 1936). This reduction in rank does shift or change some data, but also allows us to transpose our data accurately into a lower-dimensional space, thus removing the unneeded dimensions, reducing noise, and mitigating the curse of dimensionality.

As long as our training data D has been properly centered in preprocessing, we are able to transpose our training, development, and testing data into z-dimensional space by right-side multiplying each data matrix by the truncated matrix V_z for each z. To calculate our hyperparameters empirically, we used 10-fold cross-validation on the training set for each value of z from 1 to 63, the maximum value. Using *GridSearchCV* included in *skikit-learn*, which performs an exhaustive search over specified parameters, we iterated across every odd k value between 1 and 75, to prevent ties, and the Manhattan (taxicab) and Euclidean distance metrics. Next, we calculated learning curves for both AOM and MOM to evaluate the fit of the models, and finally we calculated a prediction using the testing data.

Our final prediction model was boosted random forests using AdaBoost (Freund & Schapire, 1997). Random forests are based on decision trees, which are machine learning models

that are easy to interpret and read, but may suffer from high variance and low accuracy. A tree classifies data by starting at a root node, and splitting the data between two internal nodes. Each internal node splits the data into two more internal nodes each, and this repeats until we reach a leaf node, which is a final prediction for all data points that end at that node. Trees terminate at leaf nodes when they reach one of a variety of specified parameters, including the number of samples per node, total number of levels in a tree, or specifications about the error in each node.

Trees are trained to reduce prediction error by beginning at a root node and splitting the training set into two subsets using some variable, at some threshold. The variable and thresholds are chosen to reduce impurity or inaccuracy in the dependent variables of the training set by iterating through all variables at each possible threshold, deciding on one point of one independent variable that produces the two most "pure" data subsets, and sending each subset to its own internal node. At each internal node, the algorithm again iterates through all thresholds at each variable, and splits into the two more internal nodes that maximize the reduction in impurity. This continues until impurity is not reduced by further divisions, the maximum number of nodes is reached, or some other limiting factor is met.

Impurity can be measured in several different ways, including Gini impurity for classification trees and mean squared error for regression trees. However, since a decision tree only focuses on minimizing the impurity at each stage, it cannot revisit a split once it is made. As a result, the final tree is not guaranteed to be an optimal solution (Breiman et al., 1984). Because of this and the potential for high variance, decision trees are rarely used by themselves. Instead, it is much more common to use random forests.

In a random forest (Ho, 1995), a much larger number of simpler trees are constructed. Each tree is only given access to a randomly selected group of independent variables, not the whole set, and the tree is built from a sample drawn with replacement from the training set. Because of this, each tree will be different from the others in the forest, but will probably not be as accurate as a single, full tree. These different trees will significantly decrease the variance of the prediction models, and will tend to overfit data less than individual trees. Final predictions are made by averaging their probabilistic predictions.

Still, random forests may not be sufficiently accurate in predicting some instances. To solve this problem, we used AdaBoost (Freund & Schapire, 1997). AdaBoost fits a sequence of weak predictors—we will use decision trees—on repeatedly modified versions of the training set using an iteratively-constructed random forest. The model first constructs a decision tree that predicts the dependent value for every unknown individual, all weighted evenly. The model then evaluates the performance of that individual tree, and increases the weight of individuals who were predicted poorly while decreasing the weight of individuals who were predicted accurately. A second tree is constructed based on the weighted data, then evaluated, and the weights are again adjusted. This repeats for a set number of decision trees, as subsequent trees focus on more and more difficult cases. The resultant boosted random forest makes predictions based on the average of the probabilistic predictions of the trees.

To make our final prediction models, we again used *GridSearchCV* with a 10-fold cross-validation to empirically optimize the correct hyperparameters, including the number of trees included in the forest, the manner with which we update the weights, and the contributions of each successive tree. Using these hyperparameters, we again produced learning curves demonstrating performance on the development set, training with 5% increments of the training set. Finally, we produced predictions on the testing sets and evaluated them.

Availability of Data, Code, and Materials

Python code is available in Appendix B. Individual midshipman data, however, cannot be shared, due to Human Research Protection Program protocols and information security requirements.

Results

The three prediction methods performed relatively similarly, with elastic net producing marginally better predictions on AOM and random forests producing marginally better predictions on MOM.

All models significantly outperformed random models. In 100,000 random tests, the average root mean squared error was 412.80, and the minimum was 339.41, putting the p-value for each model below .00001. Actual prediction root mean squared errors are visible in Figure 1.

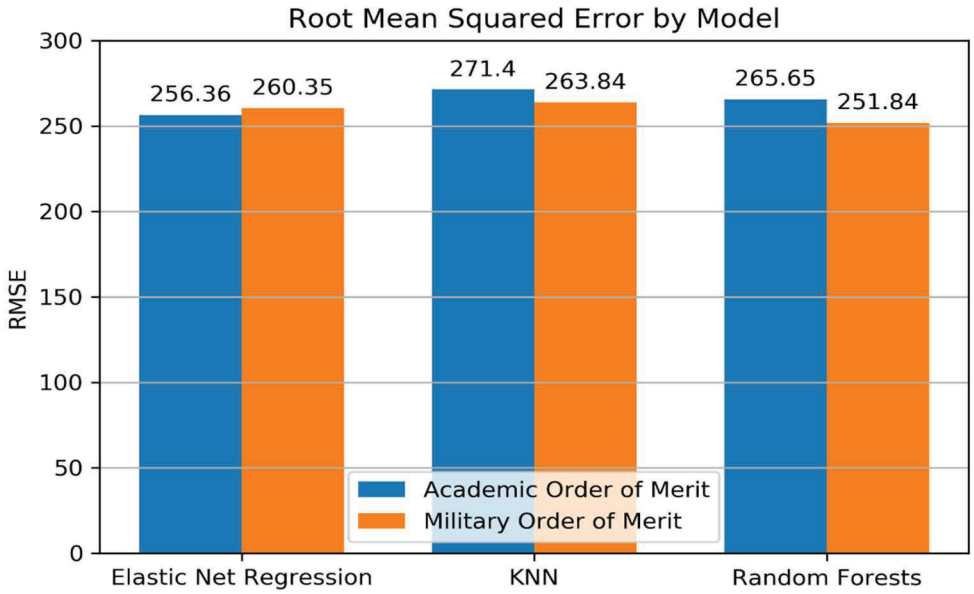


Figure 1: Root mean squared error of optimum models.

We were surprised to see the machine learning models performing at the same level as the elastic net regression, since the machine learning models are more complex and can take nonlinear factors into account. Since all models perform similarly, elastic net regression seems to be the best model, since it has the advantage of being interpretable instead of the relative black box of KNN and random forests. We will now discuss each model individually.

Elastic Net Regression. The optimum beta weights for Academic Order of Merit were found with $\alpha = 0.1$ and $L = 0.875$. The vast majority of beta weights were nonzero, potentially due to the intercorrelated nature of the predictors. The largest beta weights were Self-Efficacy (-294.43), Achievement-Striving (-203.92), Social/Personal/Emotional Intelligence (196.87), Trust (-171.14), and Modesty (168.60). The intercept for AOM was 775.56. Please note that a negative beta weight means that an increase in that trait would lead to a decrease in order of merit, which means the individual midshipman performed better.

The learning curve for elastic net regression for AOM in Figure 2 appears to be slightly overfit, in that the training set continues to outperform the development set as the curve progresses. This implies that the model could potentially benefit from either the introduction of more variables or a better understanding of nonlinear relationships.

The optimum beta weights for Military Order of Merit were found with $\alpha = 1$ and $L = 0.985$. Again, the majority of beta weights were nonzero, but a larger number were forced to zero than the weights for AOM. The largest beta weights included Trust (-187.47), Achievement-Striving (-168.85), Self-Efficacy (-158.11), Industry/Perseverance/Persistence (-155.27), and Modesty (145.84). The intercept for MOM was 934.05.

The learning curve in Figure 1 for MOM did not imply either an overfit or an underfit situation. The model seemed to capture the majority of the variance available, and neither more data nor more variables would probably be helpful to the model.

Complete beta weights for both AOM and MOM are included in Appendix C.

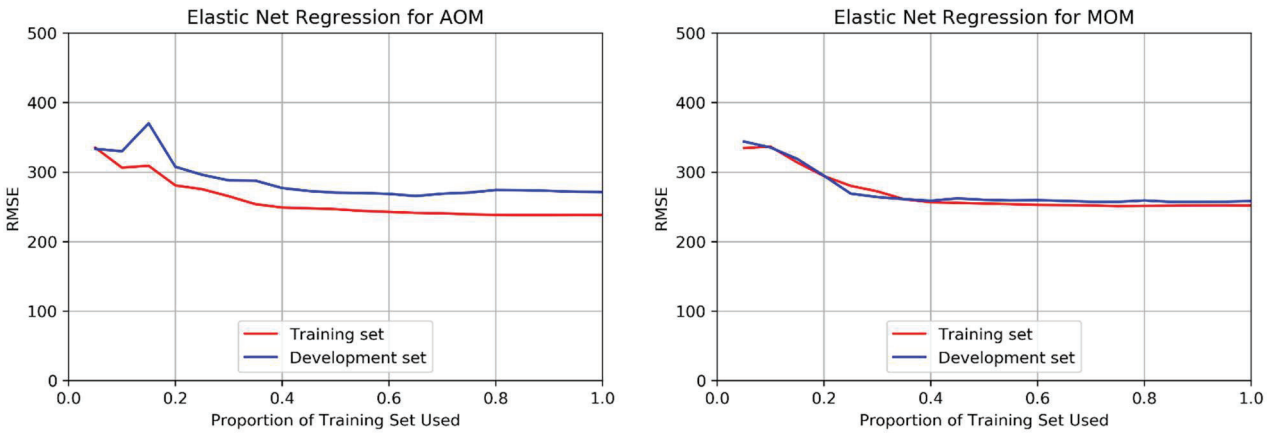


Figure 2: Learning curves for Elastic Net Regression.

KNN. The best KNN algorithm for predicting AOM was found with the singular value decomposition truncated at $z = 30$. At that level, the best algorithm was found using Euclidian distance and with $k = 11$. The optimum KNN algorithm for MOM was found with the SVD truncated at $z = 32$ and with $k = 29$. This algorithm used Euclidian distance as well. The similarity of z values implies that both AOM and MOM rely on a similar projection of our 63 independent variables into subdimensional space.

Neither learning curve in Figure 3 seems to be either overfit or underfit, although both learning curves possibly have marginally negative slopes throughout the entire graph instead of leveling off. While this potentially implies that more data could be useful, the very small magnitude of the slope suggests that any gain in prediction accuracy from adding more data would more likely than not be trivial.

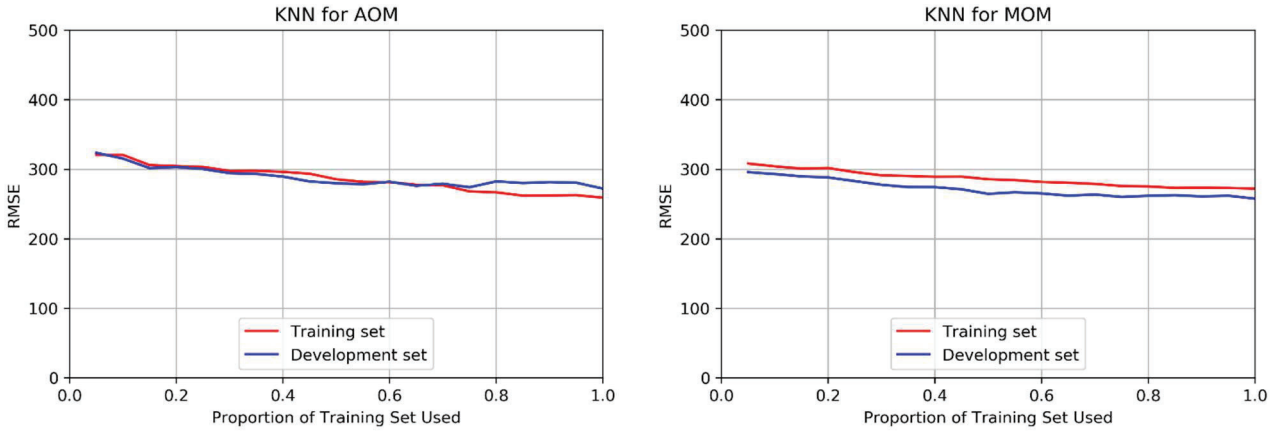


Figure 3: Learning curves for KNN.

Random Forests. The optimum boosted random forest for predicting AOM was a forest of 150 trees and a learning rate of 1. Each tree with a maximum depth of three nodes. The best boosted random forest for predicting MOM was a forest of 150 trees with a learning rate of 0.25. Each tree also had a maximum depth of three nodes.

Both random forest models seem to be overfit, visible in the growing gap between the training and development curves in Figure 4. This implies that both models were more complicated than needed, and thus trained itself on only the data and noise in the training set, not the underlying trends. If we were to make more boosted random forests, we would focus on algorithms with a smaller number of trees, less complex trees, or both.

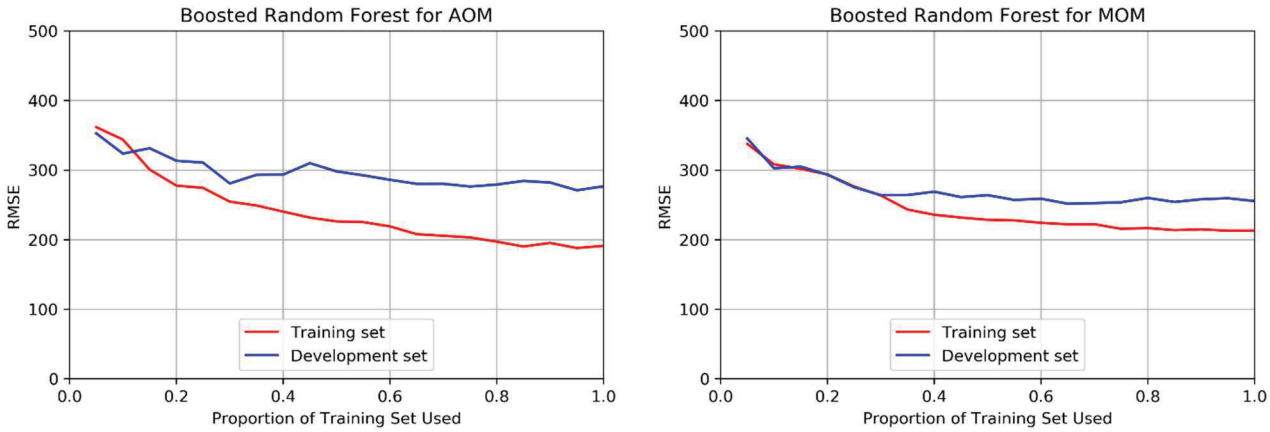


Figure 4: Learning curves for boosted random forests.

Comparative Accuracy. To judge the accuracy of our predictions, as well as to demonstrate the uncertainty in predicting our dependent variables, we constructed two final iterative models. Both are linear models, but in addition to considering psychological variables, the model progressively was allowed to access more and more information about a midshipman's performance while he or she was at the Naval Academy. Both AOM and MOM predictions became more accurate as more previous performance data was considered, but neither came close to perfection, even when the model was able to take all 63 personality traits and seven semesters of previous performance in an attempt to predict the result of the final 8th. The accuracy of these models is visible in Figure 5.

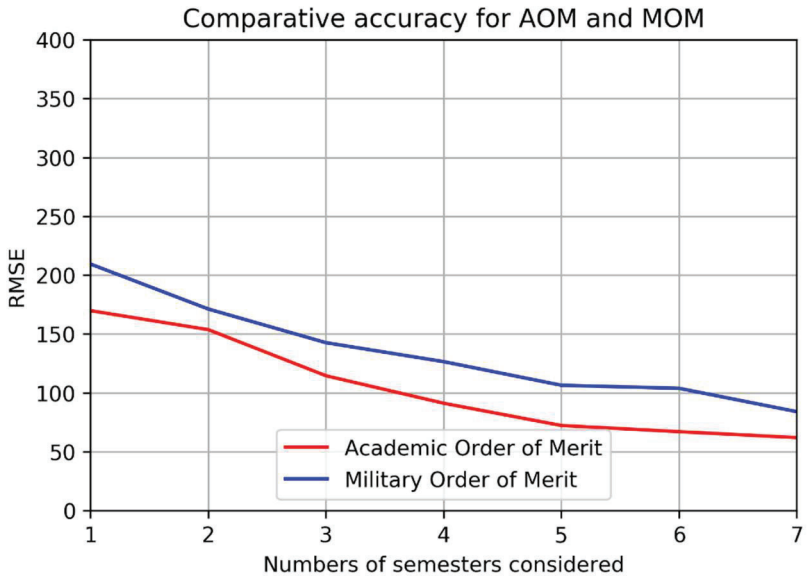


Figure 5: Iterative linear models for comparison.

Discussion

Our results are a bit mixed. On one hand, all models performed at a relatively acceptable level, beating random and average predictions consistently. On the other hand, the machine learning models failed to significantly outperform penalized regression models. Since they are more computationally complex and can take nonlinear factors into account, we expected our machine learning models to perform significantly better.

There are some significant advantages, however, to accepting elastic net regression models as potentially superior. The main one comes from interpretability. KNN and random forests are mostly "black box" algorithms, in that it is very hard to understand which variables are accounting for the most variance, and are therefore the most important. Our penalized regression models, however, each return beta weights, and further work could be done to find estimates for significance.

Another interesting conclusion comes from the rank of the singular value decompositions used to project our data into subdimensional spaces for KNN. Both of our z values—30 for AOM and 32 for MOM—are extraordinarily close to the total number of Big Five variables in our models. While we are unable to immediately see which variables are being preserved upon projection into 30- or 32-dimensional space, it does seem promising that the model matches with the most scientifically accepted model of personality.

Our iterative models for comparison, visible in Figure 5, were constructed after we were somewhat disappointed by the results of our machine learning models. We were curious on how accurate predictions could be, and if there was a point at which predictions stopped gaining significant accuracy—or if that point was never reached. Models would be 50 to 100 places off of predicting final orders of merit, even if there was only one semester of unknown performance. While this is a significant improvement over our previous models, it is much simpler to predict performance based on performance, and these models have a much smaller practical application.

Institutional Significance

While the goal of the project was never to create a tool or deliverable for use in admissions or personnel management, further research into this field looks promising. This tool could be unique to the Naval Academy, in that it would not focus on "pruning" out the lowest performers, but rather attempt to implement programs to improve performance.

The recommended goal—improving, not pruning—is recommended due to the unique situation at the Naval Academy. Similar to medical school, where the lowest graduating student still receives a M.D., all USNA graduates commission as ensigns or second lieutenants in the U.S. Navy or Marine Corps, regardless of class rank. As discussed earlier, current policy seems

to be that any individual capable enough to be admitted to USNA is capable of graduating—if they desire it. Personality-based tools seem to be capable of identifying those who may need more motivation or assistance avoiding counterproductive behaviors.

If a program was implemented to improve the performance of low-performing individuals based on personality, success could be measured by the inability to reproduce this study after a period of implementation. If personality's impact on disparate performance is reduced, further researchers would be unable to see connections as firm as ours in future classes of midshipmen.

This, however, brings us into an area of extreme caution. No tool proposed in this paper is intended to be used as an admissions or personnel management resource. There have been recent instances of machine learning models that have been implemented without proper prior consideration or transparency in several public sector fields (Rudin, Wang, & Coker, 2020). Careful ethics and Human Resource Protection Program discussions would need to be conducted before any tool begins development at USNA, and any such completed tool would need to be carefully evaluated before it could be used to make any individual, actionable prediction.

Potential New Research Developments

We believe further study into the combined dimensionality of personality across multiple tests to be very promising. Further examination into the 30- and 32-dimensional subspaces that produced the most accurate KNN predictions seems to be the next logical step. Another study could attempt to discern which exact linear combinations of facets and values make up the coordinate system in the subspaces, and attempt to measure how much variance each personality test accounts for in each subspace. Additionally, elastic net regression and boosted random forest models could be run in the subdimensional spaces, with potentially improved performances.

We also only focused on implementing KNN and boosted random forests in comparison to elastic net regression. Other machine learning models, such as clustering algorithms and neural networks warrant further investigation and comparison.

Finally, our study and both of the two listed immediately above could be completed using item-level data. While this would dramatically increase the dimensionality of our data set, item-level data can yield significantly better results in both predictions and explanations (Oswald & Hough, 2011).

Limitations

We are able to identify five main limitations in our research, although there are surely more. First, we are limited in our study to Naval Academy midshipmen, who are in no means representative of the total population. Women are significantly underrepresented, all subjects are between 17 and 25 years old, and it would be incorrect to assume that volunteers for service academies share the same personality distributions as the general population—in fact, it probably differs significantly. Second, our performance metrics are less than perfect. We were measuring a midshipman's performance as a midshipman, but the Naval Academy's mission is not to create perfect midshipmen—it is to graduate leaders dedicated to a career of naval service. If this study were to be replicated at a service academy, a better performance metric could be fitness reports in the Fleet or Marine Corps. It is more than possible to be a subpar midshipman and a successful officer. This data was unavailable to us, but future analysis could prove valuable.

Third, there was a gap of between one and a half and two and a half years between our personality tests. While the Big Five should remain relatively consistent with age, some long-term changes can be seen and predicted (Mottus & Rozgonjuk, 2021), and it is unknown if two years would be enough to impact results. The MBTI is not stable with age, and VIA is susceptible to shifts as well. It would be significantly better if the subjects took all three tests at the same time. Unfortunately, since we used previously gathered data, this was not possible for us.

Fourth, our entire study took place in a series of strong situations. Strong situations are characterized by the presence of cues from environmental forces which tell of the desirability of certain behaviors or actions. USNA is very much a strong situation—incoming midshipmen are briefed for hours on the conduct system, Midshipmen Regulations, the Honor Concept, the aptitude system, and how each class year is supposed to behave. A similar study in a much "weaker" but otherwise similar environment, in which an institution did not directly desire a particular result could yield interesting results.

Fifth, we did not use item level data for our personality tests. Research has shown that item level data produces more accurate results (Mottus & Rozgonjuk, 2021), but we were unable to access the data for two of the three tests. To avoid dominating the dimensionality with the one test, we opted to use data at one level above the item level: the 30 facets of the Big Five, the 24 values in VIA, and the four aspects of MBTI. Future studies could prove more successful if they were able to access all of that data.

Conclusion

While we were disappointed our machine learning models failed to significantly outperform penalized regression models, they presented several valuable insights into personality models, including possible suggestions of the optimal dimensionality for which to consider personality predictions. Additionally, the acceptance of the elastic net as the most powerful considered model has several advantages, including interpretability and computational cost. Further applications of machine learning models to the study of personality look promising.

References

Carter, W. E. (2017, August). USNA INSTRUCTION 1531.51B. Maryland. From https://www.usna.edu/AdminSupport/Inst/1000-1999/USNAINST%201531.51B%20Class%20Standings.pdf

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21-27.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 119-139.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 1-22.

Goldberg, L. R. (1999). A broad-bandwidth, public-domain personality inventory measuring the lower-level facets of several Five-Factor models. *Personality Psychology in Europe*, 7-28.

Hinrichs, A., Novak, E., & Wozniakowski, H. (2011). The Curse of Dimensionality for Monotone and Convex Functions of Many Variables. *Journal of Approximation Theory*, 955-965.

Ho, T. (1995). Random Decision Forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 278-282.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 55-67.

Johnson, J. A. (2014). Measuring thirty facets of the Five Factor Model with a 120-item public. *Journal of Research in Personality*, 78-89.

Kern, M. L., & Bowling, D. S. (2015). Character strengths and academic performance in law students. *Journal of Research in Personality*, 25-29.

Kim, M.-J., Park, K.-P., Seo, D.-G., & Ihm, J.-J. (2014). The relationship between dental graduate students' MBTI types and academic achievement in problem-based learning. *Korean Journal of Medical Education*, 291-297.

Kluyver, T., Ragan-Kelly, B., Perez, F., Granger, B., Bussonnier, M., Frederic, J., . . . Jupyter development team. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87-90.

Lado, M., & Alonso, P. (2017). The Five-Factor model and job performance in low complexity jobs: A quantitative synthesis. *Journal of Work and Organizational Psychology*, 175-182.

Lounsbury, J. W., Fisher, L. A., Levy, J. J., & Welsh, D. P. (2009). Investigation of character strengths in relation to the academic success of college students. *Individual Differences Research*, 52-69.

Mottus, R., & Rozgonjuk, D. (2021). Development Is in the Details: Age Differences in the Big Five Domains, Facets, and Nuances. *Journal of Personality and Social Psychology*, 1035-1048.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* Cambridge: The MIT Press.

Myers, I. B., McCaulley, M. H., Quenk, N. L., & Hammer, A. L. (1998). *MBTI Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator.* Palo Alto: Consulting Psychologists Press, Inc.

Oswald, F. L., & Hough, L. M. (2011). Personality and its Assessment in Organizations: Theoretical and Empirical Developments. In *APA Handbook of Industrial and Organizational Psychology, Volume 2* (pp. 153-174). Washington, DC: American Psychological Association.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825-2830.

Peterson, C., & Seligman, M. E. (2004). *Character Strengths and Virtues: A Handbook and Classification.* Washington, DC: American Psychological Association and Oxford University Press.

Prestov, V. (2013). Is the k-NN classifier in high dimensions affected by the curse of dimensionality? *Computers & Mathematics with Applications*, 1427-1437.

Rudin, C., Wang, C., & Coker, B. (2020). The Age of Secrecy and Unfairness in Recidivism Prediction. *Harvard Data Science Review*, 1.

Salgado, J. F. (2003). Predicting job performance using FFM and non-FFM personality measures. *Journal of Occupational and Organizational Psychology*, 323-346.

Staats, E. B. (1976). *Student Attrition At The Five Federal Service Academies .* Washington, DC: Departments of Defense, Commerce, and Transportation.

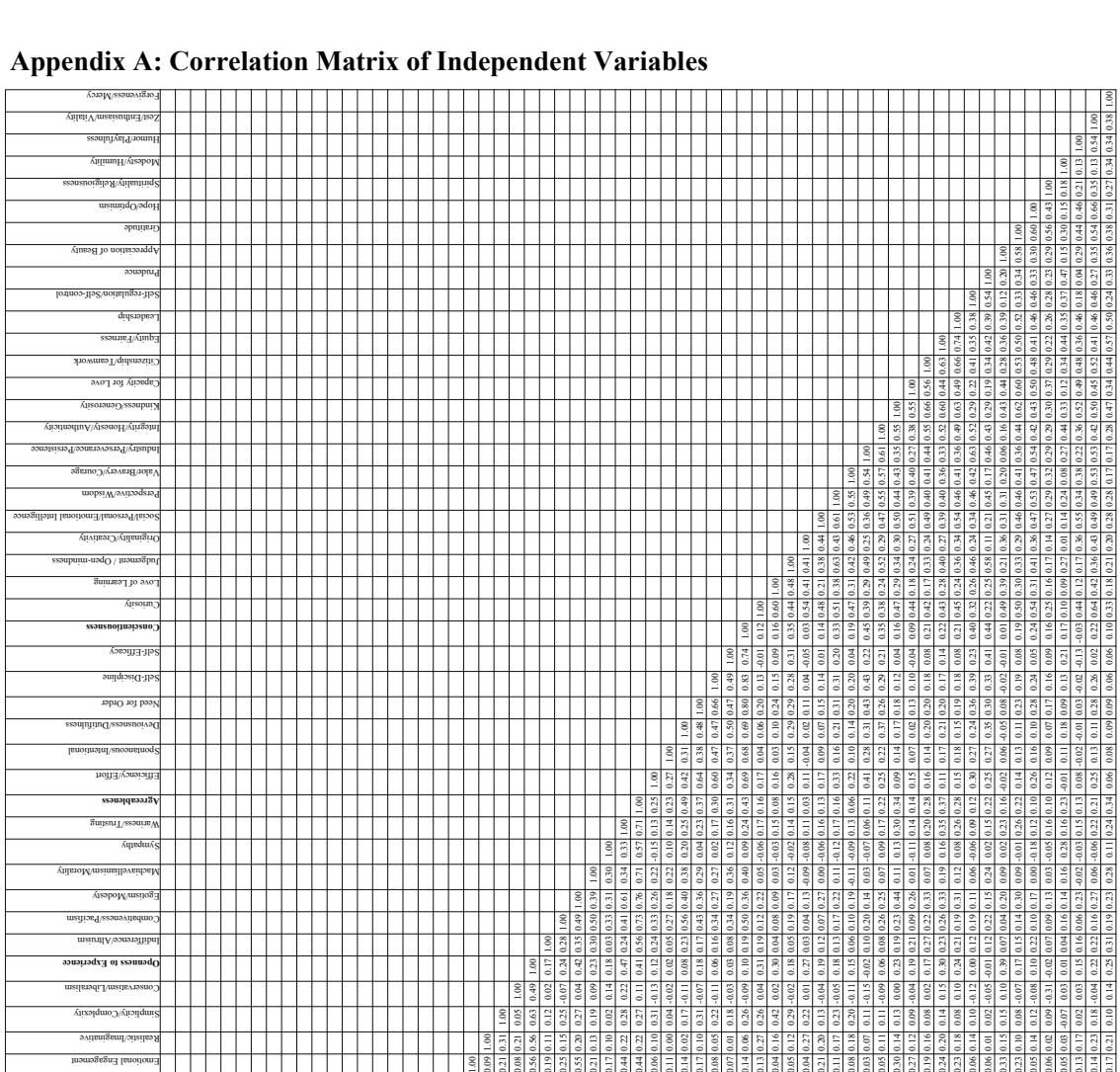
Tinshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society B*, 267-288.

Waldmann, P., Meszaros, G., Gredler, B., Fuerst, C., & Solkner, J. (2013). Evaluation of the Lasso and the Elastic Net in Genome-Wide Association Studies. *Frountiers in Genetics*, 270.

Yarkoni, T., & Westfall, J. (2017). Choosing Prediction over Explanation in Psychology: Lessons from Machine Learning. *Perspectives on Psychological Science: a journal of the Association for Psychological Science*, 1100-1122.

Zou, H., & Hastie, T. (2004). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B*, 301-320.

Appendix A: Correlation Matrix of Independent Variables



Appendix B: Jupyter Code

```
import sklearn
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import AdaBoostRegressor
from sklearn import tree
from sklearn.tree import export_graphviz
import sklearn.model_selection
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, precision_score, recall_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import ElasticNetCV
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error as MSE
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, make_scorer, r2_score
from mlxtend.evaluate import permutation_test
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 300
from matplotlib.pyplot import figure
import array as ar
import numpy as np
import pandas as pd
import seaborn as sns
import scipy
from sklearn.utils import shuffle
import random
import warnings
import csv
warnings.simplefilter('ignore', FutureWarning)
my_seed = 2
random.seed(my_seed)
np.random.seed(my_seed)
data = pd.read_excel (r'C:\Users\m216270\Desktop\Trident\Classifier and Regressor\Python Input.xlsx', sheet_name='Basic Out')
-----------------------------------------------------------------------------------------------------------------------------------------------------------
D=data.iloc[:, 0:13]
X=data.iloc[:, 13:76]
Yfull=data.iloc[:, 76:87]
plebedata=data.iloc[:,87:89]
D=shuffle(D, random_state=my_seed)
X=shuffle(X, random_state=my_seed)
Yfull=shuffle(Yfull, random_state=my_seed)
plebedata=shuffle(plebedata, random_state=my_seed)
-----------------------------------------------------------------------------------------------------------------------------------------------------------
mom=Yfull["1C SPRING MOM"]
```

```
aom=Yfull["1C SPRING AOM"]
XY=pd.concat([X, aom, mom], axis=1, sort=False)
XY=XY.dropna()
X=XY.iloc[:, :-2]
aom=XY.iloc[:, -2]
mom=XY.iloc[:,-1]
```

```
p=20 #points in the learning curve
learning_curve_x=np.linspace(1/p,1,p)
x_fig=5 #learning curve size
y_fig=3
```

```
intset_x, Xtest = sklearn.model_selection.train_test_split(X, test_size=.2, shuffle=False)
Xtrain, Xdev = sklearn.model_selection.train_test_split(intset_x, test_size=.25, shuffle=False)
intset_aom, AOMtest = sklearn.model_selection.train_test_split(aom, test_size=.2, shuffle=False)
AOMtrain, AOMdev = sklearn.model_selection.train_test_split(intset_aom, test_size=.25, shuffle=False)
intset_mom, MOMtest = sklearn.model_selection.train_test_split(mom, test_size=.2, shuffle=False)
MOMtrain, MOMdev = sklearn.model_selection.train_test_split(intset_mom, test_size=.25, shuffle=False)
```

```
zscaler=MinMaxScaler()
zscaler.fit(Xtrain)
zXtrain=zscaler.transform(Xtrain)
zXdev=zscaler.transform(Xdev)
zXtest=zscaler.transform(Xtest)
for i in range(Xtrain.shape[0]):
    for j in range(Xtrain.shape[1]):
        Xtrain.iloc[i,j]=zXtrain[i,j]
for i in range(Xdev.shape[0]):
    for j in range(Xdev.shape[1]):
        Xdev.iloc[i,j]=zXdev[i,j]
        Xtest.iloc[i,j]=zXtest[i,j]
```

Elastic Linear Regression

AOM

```
cv=RepeatedKFold(n_splits=10, n_repeats=5, random_state=my_seed)
ratios=np.arange(0,1,.005)
alphas=[1e-3, 1e-2, 1e-1, 0.0, 1.0, 10.0, 100.0]
aom_elastic_grid=ElasticNetCV(l1_ratio=ratios, alphas=alphas, cv=cv, n_jobs=-1)
aom_elastic_grid.fit(Xtrain, AOMtrain)
aom_alpha=aom_elastic_grid.alpha_
aom_l1 = aom_elastic_grid.l1_ratio_
aom_elastic_model=ElasticNet(alpha=aom_alpha, l1_ratio=aom_l1)
aom_elastic_model.fit(Xtrain, AOMtrain)
aom_predict=aom_elastic_model.predict(Xdev)
error_vector=aom_predict-AOMdev
predict_rmse = pow(    sum(pow(error_vector,2)) / Xdev.shape[0],        .5 )
predict_rmse
```

```
print(aom_alpha)
print(aom_l1)
```

```
aom_ela_learning_curve_train=np.zeros(p)
aom_ela_learning_curve_dev=np.zeros(p)
for i in range(p):
    T=Xtrain.shape[0]
    aom_ela_model=ElasticNet(alpha=aom_alpha, l1_ratio=aom_l1)
    L=((1+i)/p)*T
    L=round(L)
    aom_ela_model.fit(Xtrain.iloc[0:L, :], AOMtrain.iloc[0:L])
    aom_train_predict=aom_ela_model.predict(Xtrain)
```

```
    train_error_vector=aom_train_predict-AOMtrain
    train_predict_rmse = pow(   sum(pow(train_error_vector,2)) / Xtrain.shape[0],       .5 )
    aom_ela_learning_curve_train[i]=train_predict_rmse
    aom_dev_predict=aom_ela_model.predict(Xdev)
    dev_error_vector=aom_dev_predict-AOMdev
    dev_predict_rmse = pow(   sum(pow(dev_error_vector,2)) / Xdev.shape[0],       .5 )
    aom_ela_learning_curve_dev[i]=dev_predict_rmse
fig, ax=plt.subplots()
plt.plot(learning_curve_x, aom_ela_learning_curve_train, 'r', label="Training set")
plt.plot(learning_curve_x, aom_ela_learning_curve_dev, 'b', label='Development set')
ax.legend(loc="lower center")
plt.ylim([0,500])
#ax.set_ylabel("RMSE")
plt.xlim([0,1])
plt.grid()
plt.xlabel('Proportion of Training Set Used')
plt.ylabel('RMSE')
plt.title('Elastic Net Regression for AOM')
```

```
scipy.stats.spearmanr(aom_predict, AOMdev)
print(aom_elastic_model.coef_)
print(aom_elastic_model.intercept_)
aom_ela_model.get_params(deep=True)
```

MOM

```
cv=RepeatedKFold(n_splits=10, n_repeats=5, random_state=my_seed)
ratios=np.arange(0,1,.005)
alphas=[1e-3, 1e-2, 1e-1, 0.0, 1.0, 10.0, 100.0]
mom_elastic_grid=ElasticNetCV(l1_ratio=ratios, alphas=alphas, cv=cv, n_jobs=-1)
mom_elastic_grid.fit(Xtrain, MOMtrain)
mom_alpha=mom_elastic_grid.alpha_
mom_l1 = mom_elastic_grid.l1_ratio_
mom_elastic_model=ElasticNet(alpha=mom_alpha, l1_ratio=mom_l1)
mom_elastic_model.fit(Xtrain, MOMtrain)
mom_predict=mom_elastic_model.predict(Xdev)
error_vector=mom_predict-MOMdev
predict_rmse = pow(   sum(pow(error_vector,2)) / Xdev.shape[0],       .5 )
predict_rmse
```

```
print(mom_elastic_model.coef_)
print(mom_elastic_model.intercept_)
mom_elastic_model.get_params(deep=True)
```

```
mom_ela_learning_curve_train=np.zeros(p)
mom_ela_learning_curve_dev=np.zeros(p)
for i in range(p):
    T=Xtrain.shape[0]
    mom_ela_model=ElasticNet(alpha=mom_alpha, l1_ratio=mom_l1)
    L=((1+i)/p)*T
    L=round(L)
    mom_ela_model.fit(Xtrain.iloc[0:L, :], MOMtrain.iloc[0:L])
    mom_train_predict=mom_ela_model.predict(Xtrain)
    train_error_vector=mom_train_predict-MOMtrain
    train_predict_rmse = pow(   sum(pow(train_error_vector,2)) / Xtrain.shape[0],       .5 )
    mom_ela_learning_curve_train[i]=train_predict_rmse
    mom_dev_predict=mom_ela_model.predict(Xdev)
    dev_error_vector=mom_dev_predict-MOMdev
    dev_predict_rmse = pow(   sum(pow(dev_error_vector,2)) / Xdev.shape[0],       .5 )
    mom_ela_learning_curve_dev[i]=dev_predict_rmse
fig, ax=plt.subplots()
```

```
plt.plot(learning_curve_x, mom_ela_learning_curve_train, 'r', label="Training set")
plt.plot(learning_curve_x, mom_ela_learning_curve_dev, 'b', label='Development set')
ax.legend(loc="lower center")
plt.ylim([0,500])
plt.xlim([0,1])
plt.grid()
plt.xlabel('Proportion of Training Set Used')
plt.ylabel('RMSE')
plt.title('Elastic Net Regression for MOM')
```

KNN

```
scaler=StandardScaler()
scaler.fit(Xtrain)
sXtrain=scaler.transform(Xtrain)
sXdev=scaler.transform(Xdev)
sXtest=scaler.transform(Xtest)
```

AOM

```
svd_range=range(1,X.shape[1])
krange=range(1,101)
aom_best_knn=''
aom_best_rmse=np.inf
aom_zbest=0
u, s, vh = np.linalg.svd(sXtrain, full_matrices=False)
params = {
    'n_neighbors': np.arange(1,75, 2),
    'p' : [1, 2]
    }
best_k_tracker=np.zeros(X.shape[1])
rmse_tracker=np.zeros(X.shape[1])
for z in svd_range:
   U=u[:, :z]
   S=s[:z]
   Vh=vh[:z, :]
   V=np.transpose(Vh)
   svd_train=U*S
   #svd_dev=np.dot(Xdev, V)
   knn_aom=KNeighborsRegressor()
   score = make_scorer(mean_squared_error, greater_is_better=False)
   gridsearch=GridSearchCV(knn_aom, params, scoring=score, cv=10, return_train_score=True)
   gridsearch.fit(svd_train, AOMtrain)
   gridsearch.best_estimator_
   if pow(-gridsearch.best_score_, .5) < aom_best_rmse:
      aom_best_rmse=pow(-gridsearch.best_score_, .5)
      aom_best_knn=gridsearch.best_estimator_
      aom_zbest=z
   print(z)
```

```
print(aom_best_knn)
print(aom_zbest)
```

```
aom_knn_learning_curve_train=np.zeros(p)
aom_knn_learning_curve_dev=np.zeros(p)
aom_k=aom_best_knn.get_params()['n_neighbors']
U=u[:, :aom_zbest]
S=s[:aom_zbest]
Vh=vh[:aom_zbest, :]
svd_train=U*S
svd_dev=np.dot(sXdev, np.transpose(Vh))
for i in range(p):
```

```
    T=svd_train.shape[0]
    aom_knn_model=aom_best_knn
    L=((1+i)/p)*T
    L=round(L)
    aom_knn_model.set_params(n_neighbors=aom_k)
    if L < aom_k:
        aom_knn_model.set_params(n_neighbors=L)
    else:
        aom_knn_model.set_params(n_neighbors=aom_k)
    aom_knn_model.fit(svd_train[0:L, :], AOMtrain.iloc[0:L])
    aom_train_predict=aom_knn_model.predict(svd_train)
    train_error_vector=aom_train_predict-AOMtrain
    train_predict_rmse = pow(   sum(pow(train_error_vector,2)) / svd_train.shape[0],        .5 )
    aom_knn_learning_curve_train[i]=train_predict_rmse
    aom_dev_predict=aom_knn_model.predict(svd_dev)
    dev_error_vector=aom_dev_predict-AOMdev
    dev_predict_rmse = pow(   sum(pow(dev_error_vector,2)) / svd_dev.shape[0],        .5 )
    aom_knn_learning_curve_dev[i]=dev_predict_rmse
    print(aom_knn_model.get_params()['n_neighbors'])
fig, ax=plt.subplots()
plt.plot(learning_curve_x, aom_knn_learning_curve_train, 'r', label="Training set")
plt.plot(learning_curve_x, aom_knn_learning_curve_dev, 'b', label='Development set')
ax.legend(loc="lower center")
plt.ylim([0,500])
plt.xlim([0,1])
plt.grid()
plt.xlabel('Proportion of Training Set Used')
plt.ylabel('RMSE')
plt.title('KNN for AOM')
```

MOM

```
svd_range=range(1,X.shape[1])
krange=range(1,101)
mom_best_knn=''
mom_best_rmse=np.inf
mom_zbest=0
u, s, vh = np.linalg.svd(sXtrain, full_matrices=False)
params = {
    'n_neighbors': np.arange(1,75, 2),
    'p' : [1, 2]
    }
best_k_tracker=np.zeros(X.shape[1])
rmse_tracker=np.zeros(X.shape[1])
for z in svd_range:
    U=u[:, :z]
    S=s[:z]
    Vh=vh[:z, :]
    V=np.transpose(Vh)
    svd_train=U*S
    #svd_dev=np.dot(Xdev, V)
    knn_mom=KNeighborsRegressor()
    score = make_scorer(mean_squared_error, greater_is_better=False)
    gridsearch=GridSearchCV(knn_mom, params, scoring=score, cv=10, return_train_score=True)
    gridsearch.fit(svd_train, MOMtrain)
    gridsearch.best_estimator_
    if pow(-gridsearch.best_score_, .5) < mom_best_rmse:
        mom_best_rmse=pow(-gridsearch.best_score_, .5)
        mom_best_knn=gridsearch.best_estimator_
        mom_zbest=z
    print(z)
```

```
print(mom_best_knn)
print(mom_zbest)
```

```
mom_knn_learning_curve_train=np.zeros(p)
mom_knn_learning_curve_dev=np.zeros(p)
mom_k=mom_best_knn.get_params()['n_neighbors']
U=u[:, :mom_zbest]
S=s[:mom_zbest]
Vh=vh[:mom_zbest, :]
svd_train=U*S
svd_dev=np.dot(sXdev, np.transpose(Vh))
for i in range(p):
    T=svd_train.shape[0]
    mom_knn_model=mom_best_knn
    L=((1+i)/p)*T
    L=round(L)
    mom_knn_model.set_params(n_neighbors=mom_k)
    if L < mom_k:
        mom_knn_model.set_params(n_neighbors=L)
    else:
        mom_knn_model.set_params(n_neighbors=mom_k)
    mom_knn_model.fit(svd_train[0:L, :], MOMtrain.iloc[0:L])
    mom_train_predict=mom_knn_model.predict(svd_train)
    train_error_vector=mom_train_predict-MOMtrain
    train_predict_rmse = pow(   sum(pow(train_error_vector,2)) / svd_train.shape[0],       .5 )
    mom_knn_learning_curve_train[i]=train_predict_rmse
    mom_dev_predict=mom_knn_model.predict(svd_dev)
    dev_error_vector=mom_dev_predict-MOMdev
    dev_predict_rmse = pow(   sum(pow(dev_error_vector,2)) / svd_dev.shape[0],       .5 )
    mom_knn_learning_curve_dev[i]=dev_predict_rmse
    print(mom_knn_model.get_params()['n_neighbors'])
fig, ax=plt.subplots()
plt.plot(learning_curve_x, mom_knn_learning_curve_train, 'r', label="Training set")
plt.plot(learning_curve_x, mom_knn_learning_curve_dev, 'b', label='Development set')
ax.legend(loc="lower center")
plt.ylim([0,500])
plt.xlim([0,1])
plt.grid()
plt.xlabel('Proportion of Training Set Used')
plt.ylabel('RMSE')
plt.title('KNN for MOM')
```

Random Forests

AOM

```
ada_aom=AdaBoostRegressor()
params = {
 'n_estimators': [50, 100, 150, 200],
 'learning_rate' : [0.01, 0.05, 0.1, 1],
 'loss' : ['linear', 'square', 'exponential']
 }
score = make_scorer(mean_squared_error)
gridsearch=GridSearchCV(ada_aom, params, cv=10, return_train_score=True)
gridsearch.fit(Xtrain, AOMtrain)
aom_best_ada=gridsearch.best_estimator_
print(aom_best_ada)
print(aom_best_ada.base_estimator_)
```

```
aom_best_ada.fit(Xtrain, AOMtrain)
aom_dev_predict=aom_best_ada.predict(Xdev)
mse=mean_squared_error(aom_dev_predict, AOMdev)
rmse=pow(mse, .5)
```

```
print(rmse)
```

```
aom_ada_learning_curve_train=np.zeros(p)
aom_ada_learning_curve_dev=np.zeros(p)
for i in range(p):
    T=Xtrain.shape[0]
    aom_ada_model=aom_best_ada
    L=((1+i)/p)*T
    L=round(L)
    aom_ada_model.fit(Xtrain.iloc[0:L, :], AOMtrain.iloc[0:L])
    aom_train_predict=aom_ada_model.predict(Xtrain)
    train_error_vector=aom_train_predict-AOMtrain
    train_predict_rmse = pow(   sum(pow(train_error_vector,2)) / Xtrain.shape[0],       .5 )
    aom_ada_learning_curve_train[i]=train_predict_rmse
    aom_dev_predict=aom_ada_model.predict(Xdev)
    dev_error_vector=aom_dev_predict-AOMdev
    dev_predict_rmse = pow(   sum(pow(dev_error_vector,2)) / Xdev.shape[0],       .5 )
    aom_ada_learning_curve_dev[i]=dev_predict_rmse
fig, ax=plt.subplots()
plt.plot(learning_curve_x, aom_ada_learning_curve_train, 'r', label="Training set")
plt.plot(learning_curve_x, aom_ada_learning_curve_dev, 'b', label='Development set')
ax.legend(loc="lower center")
plt.ylim([0,500])
plt.xlim([0,1])
plt.grid()
plt.xlabel('Proportion of Training Set Used')
plt.ylabel('RMSE')
plt.title('Boosted Random Forest for AOM')
```

```
# P value calculations for AOM
treatment=abs(aom_dev_predict-AOMdev)
# other control options
# control=shuffle(treatment, random_state=my_seed)
I = [i+1 for i in range(len(aom))]
random.shuffle(I)
I = I[:AOMdev.size]
control=Ip_value=permutation_test(treatment, control, method='approximate', num_rounds=10000, seed=my_seed)
print(p_value)
```

MOM

```
ada_mom=AdaBoostRegressor()
params = {
 'n_estimators': [50, 100, 150, 200],
 'learning_rate' : [0.01, 0.1, .25, 0.5, 1],
 'loss' : ['linear', 'square', 'exponential']
 }
score = make_scorer(mean_squared_error)
gridsearch=GridSearchCV(ada_mom, params, cv=10, return_train_score=True)
gridsearch.fit(Xtrain, MOMtrain)
mom_best_ada=gridsearch.best_estimator_
print(mom_best_ada)
print(mom_best_ada.base_estimator_)
```

```
mom_best_ada.fit(Xtrain, MOMtrain)
mom_dev_predict=mom_best_ada.predict(Xdev)
mse=mean_squared_error(mom_dev_predict, MOMdev)
rmse=pow(mse, .5)
print(rmse)
```

```
# P value calculations for MOM
```

```
treatment=abs(mom_dev_predict-MOMdev)
# other control options
# control=shuffle(treatment, random_state=my_seed)
I = [i+1 for i in range(len(mom))]
random.shuffle(I)
I = I[:MOMdev.size]
control=I
p_value=permutation_test(treatment, control, method='approximate', num_rounds=10000, seed=my_seed)
print(p_value)
scipy.stats.spearmanr(mom_dev_predict, MOMdev)
```

```
mom_ada_learning_curve_train=np.zeros(p)
mom_ada_learning_curve_dev=np.zeros(p)
for i in range(p):
    T=Xtrain.shape[0]
    mom_ada_model=mom_best_ada
    L=((1+i)/p)*T
    L=round(L)
    mom_ada_model.fit(Xtrain.iloc[0:L, :], MOMtrain.iloc[0:L])
    mom_train_predict=mom_ada_model.predict(Xtrain)
    train_error_vector=mom_train_predict-MOMtrain
    train_predict_rmse = pow(    sum(pow(train_error_vector,2)) / Xtrain.shape[0],        .5 )
    mom_ada_learning_curve_train[i]=train_predict_rmse
    mom_dev_predict=mom_ada_model.predict(Xdev)
    dev_error_vector=mom_dev_predict-MOMdev
    dev_predict_rmse = pow(    sum(pow(dev_error_vector,2)) / Xdev.shape[0],        .5 )
    mom_ada_learning_curve_dev[i]=dev_predict_rmse
fig, ax=plt.subplots()
plt.plot(learning_curve_x, mom_ada_learning_curve_train, 'r', label="Training set")
plt.plot(learning_curve_x, mom_ada_learning_curve_dev, 'b', label='Development set')
ax.legend(loc="lower center")
plt.ylim([0,500])
plt.xlim([0,1])
plt.grid()
plt.xlabel('Proportion of Training Set Used')
plt.ylabel('RMSE')
plt.title('Boosted Random Forest for MOM')
```

```
aom_ela_test_rmse=pow(MSE(aom_ela_model.predict(Xtest), AOMtest), .5)
mom_ela_test_rmse=pow(MSE(mom_ela_model.predict(Xtest), MOMtest), .5)
aom_ada_test_rmse=pow(MSE(aom_ada_model.predict(Xtest), AOMtest), .5)
mom_ada_test_rmse=pow(MSE(mom_ada_model.predict(Xtest), MOMtest), .5)
```

```
u, s, vh = np.linalg.svd(sXtrain, full_matrices=False)
U=u[:, :aom_zbest]
S=s[:aom_zbest]
Vh=vh[:aom_zbest, :]
svd_train=U*S
svd_test=np.dot(sXtest, np.transpose(Vh))
aom_knn_model=aom_best_knn
aom_knn_model.set_params(n_neighbors=aom_k)
aom_knn_model.fit(svd_train, AOMtrain)
aom_knn_test_rmse=pow(MSE(aom_knn_model.predict(svd_test), AOMtest), .5)
```

```
u, s, vh = np.linalg.svd(sXtrain, full_matrices=False)
U=u[:, :mom_zbest]
S=s[:mom_zbest]
Vh=vh[:mom_zbest, :]
svd_train=U*S
svd_test=np.dot(sXtest, np.transpose(Vh))
```

```
mom_knn_model=mom_best_knn
mom_knn_model.set_params(n_neighbors=mom_k)
mom_knn_model.fit(svd_train, MOMtrain)
mom_knn_test_rmse=pow(MSE(mom_knn_model.predict(svd_test), MOMtest), .5)
```

```
labels=['Elastic Net Regression',
       'KNN',
       'Random Forests']
aom_scores=[aom_ela_test_rmse,
          aom_knn_test_rmse,
          aom_ada_test_rmse]
mom_scores=[mom_ela_test_rmse,
          mom_knn_test_rmse,
          mom_ada_test_rmse]
x=np.arange(len(labels))
width=.35
fig, ax=plt.subplots()
plt.grid(axis='y')
plt.ylim([0,300])
rects1 = ax.bar(x - width/2, aom_scores, width, label='Academic Order of Merit')
rects2 = ax.bar(x + width/2, mom_scores, width, label='Military Order of Merit')
ax.set_ylabel('RMSE')
ax.set_title('Root Mean Squared Error by Model')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()
for rect, label in zip(rects1, aom_scores):
   height = rect.get_height()
   ax.text(rect.get_x() + rect.get_width() / 2, height + 5, round(label,2),
          ha='center', va='bottom')
for rect, label in zip(rects2, mom_scores):
   height = rect.get_height()
   ax.text(rect.get_x() + rect.get_width() / 2, height + 5, round(label,2),
          ha='center', va='bottom')
ax.legend(loc='lower center')
plt.figure(figsize=(10,10))
fig.tight_layout()
plt.show()
```

```
print(aom_scores)
print(mom_scores)
r=np.zeros(100000)
for i in range(100000):
   rand_X=shuffle(AOMtest, random_state=i)
   r[i]=pow(MSE(AOMtest, rand_X), .5)
print(np.mean(r))
print(min(r))
avg_X=np.zeros(AOMtest.shape)+np.mean(AOMtrain)
avg_rmse=pow(MSE(AOMtest, avg_X), .5)
avg_rmse
```

Appendix C: Table of Beta Weights for Elastic Net Regression

Trait/Facet	AOM Beta Weight	MOM Beta Weight
INTERCEPT	775.56	934.05
Extraversion/Introversion	29.25	15.46
Sensing/Intuitive	-26.84	-6.33
Thinking/Feeling	37.07	0.00
Judging/Perceiving	-75.53	-48.00
Openness to Experience	-4.89	0.00
Imagination (O1)	66.58	73.30
Artistic Interests (O2)	26.50	0.00
Emotionality (O3)	-28.36	-15.70
Adventurousness (O4)	37.63	-1.23
Intellect (O5)	-125.90	-32.86
Liberalism (O6)	-27.72	0.00
Conscientiousness	-73.79	-2.97
Self-Efficacy (C1)	-294.43	-158.11
Orderliness (C2)	109.87	68.56
Dutifulness (C3)	30.47	0.00
Achievement-Striving (C4)	-203.92	-168.85
Self-Discipline (C5)	28.25	0.00
Cautiousness (C6)	-128.77	-89.13
Extraversion	0.46	0.00
Friendliness (E1)	-61.48	-47.01
Gregariousness (E2)	0.00	34.67
Assertiveness (E3)	36.41	-104.61
Activity Level (E4)	-63.93	-84.16
Excitement-Seeking (E5)	0.00	0.00
Cheerfulness (E6)	113.71	0.00
Agreeableness	0.00	0.00
Trust (A1)	-171.14	-187.47
Morality (A2)	-24.78	0.00
Altruism (A3)	-17.30	-42.27
Cooperation (A4)	-6.91	-37.60
Modesty (A5)	168.60	145.84

Sympathy (A6)	42.18	0.00
Neuroticism	-23.80	0.00
Anxiety (N1)	-2.94	-41.23
Anger (N2)	22.11	0.00
Depression (N3)	-50.10	4.93
Self-Consciousness (N4)	-80.71	-27.61
Immoderation (N5)	23.16	2.51
Vulnerability (N6)	-69.76	-16.39
Curiosity	-65.88	-46.28
Love of Learning	-110.53	-60.21
Judgement/ Open-Mindedness	-75.71	0.00
Originality/Creativity	34.80	65.53
Social/Personal/Emotional Intelligence	196.87	117.42
Perspective/Wisdom	24.09	17.74
Valor/Bravery/Courage	8.48	0.00
Industry/Perseverance/ Persistence	-180.23	-155.27
Integrity/Honesty/ Authenticity	68.22	43.41
Kindness/Generosity	69.55	25.26
Capacity for Love	-4.57	0.00
Citizenship/Teamwork	76.31	3.09
Equity/Fairness	-56.05	0.00
Leadership	127.50	48.92
Self-Regulation/Self-Control	118.43	0.00
Prudence	-68.10	-59.35
Appreciation of Beauty	-2.72	43.03
Gratitude	38.81	11.17
Hope/Optimism	13.09	0.00
Spirituality/Religiousness	13.90	0.00
Modesty/Humility	136.66	49.27
Humor/Playfulness	-48.75	0.00
Zest/Enthusiasm/Vitality	21.16	0.00
Forgiveness/Mercy	-27.53	3.02