MATH 656

Professor He

Midterm **GROUP 6 | Datathon Report**

Introduction & Data Cleaning:

The dataset is organized as a large-scale test for adults called "The Programme for the International Assessment of Adult Competencies" (PIACC). The OECD organizes this test for working adults in over forty countries to measure their skills in a technology working environment. It tests adults in literacy, numeracy, and problem-solving skills. However, the problem-solving component only provides binary responses (correctness/incorrectness) and does not provide much insight for the OECD. Thus, in our class, we use data mining techniques to understand and identify Problem Solving in a Technology-Rich Environment (PSTRE). This study collects data by asking a series of questions in behavioral and background information. The test is computer-based. This method enables the organization to have a better understanding of how test takers approach the test, how long they spend on each question or the order of answered questions. The goal is to use variables in the dataset to predict the adults' PSTRE level (gold standard). Such a result will help educators, researchers, and policymakers to understand how adults solve problems and further assist countries to develop helpful policies and programs.

To provide a brief overview given this contextual understanding, the goal of the project is to apply supervised machine learning methods to this data with the aim of making predictions on the problem-solving skill levels (PS_class) of a new test-taker. As the raw data has missing values, we used two techniques to fill in missing data: "mean value" and an R package ("MICE") that helps turn missing data into clean data. After cleaning, we used feature selection, namely Learning Vector Quantization (LVQ) and embedded feature selection in Random Forest to

identify what background or behavioral variables are predictive of adult problem-solving abilities. We also pinpointed how accurate these variables are in predicting response variables. Finally, we put our selected variables into the three chosen models (KNN, Random Forest, and Naïve Bayes Classifiers) to discern which model & technique garnered the greatest accuracy.

The raw data required considerable cleaning efforts. We used the str() function in R to understand the structure of the dataset, learning facts such as it had a total of 664 respondents and 588 variables in our sample size. Additionally, it should be noted that the dataset is made of numerics and characters. For instance, one of the character variables looked like the following: "u01a_item101 |u01a_item202 | u01a_item202 |u01a_item201 | u01a_item202 | u01a_item201 | u01a_item201 | u01a_item201". As these characters did not help us predict the gold standard, we only selected the numerical variables. Now that the dataset is only comprised of non-character data, we observed that in some cells, there were erroneous values such as 0, 6, 7, 8, 9, 99, 999, 9999, & so forth. These represent skip questions, invalid answers, or refusal to answer in the non-gold standard columns. If we used these numbers, the output would have been skewed and incorrect. Thus, we treated these values as missing values (#N/A). However, the gold standard, also called PS_Class, had four values 0, 1, 2, and 3, which correspond to Below Level 1, Level 1, Level 2, and Level 3 respectively. In other columns, 0 meant missing values, in contrast, 0 is a valid data point in the gold standard. For this reason, we had to isolate the gold standard column, converted the remaining columns to NA, and then used the cbind() function to combine the converted data and the gold standard. Next, we removed any columns that have more than 20% of NA values, as 20% of 664 rows is 133 NA values, and that is a lot of missing data. As a result, the new dataset had 367 variables, including the gold standard. Next, the team filled in the missing data with the mean value calculated from the remaining data in the columns. We ran for

the first time and observed that there was a limit of 72.23% accuracy. We reasoned that if we can modify our cleaning method, we might have better accuracy. We researched and agreed on two things: filling the missing values with mean is not the optimal method and the better solution could be the MICE (Multivariate Imputation via Chained Equations) package. There are several issues with using mean value to fill in missing data. Firstly, this approach adds no new information but only increases the sample size and leads to an underestimation of the errors. [1] Secondly, the mean value has its limitation when the data has a high standard deviation or even outliers. For example, we have four data points including one missing $x_1 = 3$, $x_2 = 5$, $x_3 = ?$ and $x_4 = 100$. Using the mean, we can find $x_3 = 36$. If x_4 is an outlier, $x_3=36$ is not a valid estimate. To overcome these issues, the R package called "MICE" is recommended. MICE package uses multivariate method imputation with data from the other columns to provide the best prediction for each missing value. MICE runs regression models with the missing value as the response variable and the remaining are used as explanatory variables. Depending on the type of missing values, MICE uses different types of regression. For instance, linear regression is used to predict continuous missing values. Logistic regression is used for categorical missing values. [2]

Method:

There were several methods and models that made this project a roaring success. In terms of methods, we have already discussed the MICE package above, however, the feature selection methodology deserves most explanation. We first applied feature selection by intuition for example assuming people who use ICT skills at work and home would perform better in the assessment on problem solving skills. However, the team realized a more robust methodology was necessary as 588 variables were too many variables (data frame columns) to review

manually. Thus, the team elected to use the embedded feature selection as part of the Random Forest model to begin (results discussed later in report) as from basic research it appeared to be one of the better supervised ML algorithms and required no additional feature selection. Thus, it was our first model. Indeed, the Random Forest models "attempt to improve the generalization performance by constructing an ensemble of *decorrelated* decision trees". [3] By using the notion of bagging with a twist, namely a different bootstrap sample of the data for training, Random Forests is especially useful in how it has embedded feature selection to automatically choose the most accurate tree forking path. In addition to this method, the team attempted to rank features by importance using Learning Vector Quantization ("LVQ"). LVQ uses a competitive learning algorithm to engage in supervised machine learning; simply put, it is an artificial neural network that helped the team select the best features in the data. [4] It is helpful because it works on problems for multi-class classifications, such as our situation here, where the project parameters necessitate classifying our data into one of three categories (below level 1, level 1, level 2, level 3). The feature selection used the varlmp function from LVQ, and then the team plotted the outcome to show the top features in the dataset, such that the top 20 and then 31 most important attributes would be fed into the model in hopes of having better accuracy (one developer reported 31 features seemed to best improve the model and the 20 was the original output of the feature selection technique). [5]

Two other models were employed in addition to Random Forest, namely KNN & Naïve Bayes. KNN (k-Nearest Neighbors) uses a Euclidean distance function to assign a given new/unknown record to a class. However, it can result in bias if variables not weighted properly and lose accuracy if high dimensionality data. [6] The Naïve Bayes algorithm is robust for noise & irrelevant attributes, but it assumes independence across features which will be further discussed

later in the report as a disadvantage to the model's performance. Lastly, it should be noted that the team used the set.seed(#) command to help make the model outputs reproducible with ensuing runs of the code.

Feature Selection:

The importance of features can be estimated from data by building a model. The advantage of using a model-based approach is that it is more closely tied to the model performance and it *may* be able to incorporate the correlation structure between the predictors in the importance calculation.

At first, we chose variables by intuition. However, that method is not efficient nor necessarily the best. We then constructed a Learning Vector Quantization model, to help us separate variables into two groups: those that use the model information and those that do not. The reasons for choosing the LVQ model is that it is developed and is best understood as a classification algorithm, which support multi-class classification problems. The data science team prepared the training scheme for LVQ model with a repeated 10-fold cross-validation. As we all know, a single run of the k-fold cross-validation procedure may result in a noisy estimate of model performance. Different splits of the data may result in very different results. Repeated k-fold cross-validation provides a way for improvement.

As shown in *Figure1*, x0, x1, x2, x3 represent level 1 to level 4 in our gold standard, respectively. Each attribute has an information score on each level in the Gold Standard (PS_class). For each attribute, we took the average of the information scores on each level and assign it on the new column we create, called x_mean. Then we ranked all attributes which has the highest average information score from high to low.

	X0 <dbl>	X1 <dbl>	X2 <dbl>	X3 <dbl>	x_mean <dbl>		X0 <dbl>	X1 <dbl>	X2 <dbl>	X3 <dbl>	x_mean <dbl>
PVLIT1	97.9997285	100.000000	74.3077537	100.00000000	93.076871	PS1_u01b_num_of_diff_emails_views	73.5410652	77.228381	41.3861067	77.22838137	67.345984
PVNUM1	94.4506086	100.000000	66.9108552	100.00000000	90.340366	U06b000A	68.8261007	78.337029	40.3359386	78.33702882	66.459024
PS2_u02_time_on_task	85.0757048	94.589800	56.4813824	94.58980044	82.684172	PS1_u01a_num_of_diff_emails_views	69.7924793	78.048780	38.1094375	78.04878049	65.999869
PS2_u07_num_of_diff_visited_pages	84.0783746	96.407982	51.7410004	96.40798226	82.158835	PS1_u01a_num_of_email_views	66.0679669	75.986696	37.3411151	75.98669623	63.845619
PS1_u04a_time_on_task	76.4780307	90.243902	52.0493308	90.24390244	77.253792	PS2_u07_time_on_task	65.4317390	76.762749	34.4453789	76.76274945	63.350654
PS2_u07_num_of_page_visits	77.7092176	89.024390	45.9696881	89.02439024	75.431922	PS2_u23_time_on_task	66.3809222	72.682927	40.3788941	72.68292683	63.031417
PS1_u01b_num_of_email_views	78.3420064	84.168514	44.5013635	84.16851441	72.795100	U02x000A	65.1841260	70.909091	43.8667168	70.90909091	62.717256
PS2_u02_num_of_diff_emails_views	69.0599575	88.470067	40.6079019	88.47006652	71.651998	U01b000A	68.1382868	68.780488	42.0167345	68.78048780	61.928999
PS1_u06b_num_of_page_visits	72.7535183	81.241685	41.8491293	81.24168514	69.271504	PS2_u11b_time_on_task	63.7328386	72.372506	38.5258812	72.37250554	61.750933
U07x000A	65.8306711	84.390244	38.3515559	84.39024390	68.240679	PS1_u01b_time_on_task	65.8856962	67.893570	37.2991479	67.89356984	59.742996

Figure 1

We observed that among these 20 variables, only 2 are background variables, which are PVLIT1 (Literacy Scale Score) and PVNUM1 (Numeracy Scale Score) and the rest are behavioral variables. However, these two background variables were surprisingly the most informative variables to predict the gold standard.

Model Results and Discussion:

We used 3 different data mining models, namely, Random Forest, K-nearest neighbors and Naïve Bayes.

Random Forest:

We used 4 different variations of Random Forest model. The first three models used data with mean value cleaning method. The fourth model used data cleaned with the MICE package. Since we know that Random Forest already has an embedded feature selection, we put all the attributes (366 attributes) into our first model without using any feature selection method. We used 80% of our data for training and 20% for testing, as these proportions are often taught to be one of the better industry standards. We used 10-fold cross-validation to help us generate a less biased or less optimistic estimate of the model. Using tuneLength=365, the results are shown as *Figure 2*. The tuneLength "allows system to tune algorithm automatically. It indicates the number of different values to try for each tunning parameter. For example, mtry is such an example for randomForest. Suppose tuneLength = 5, it means try 5 different mtry values and find the optimal mtry value based on these 5 values." [7]

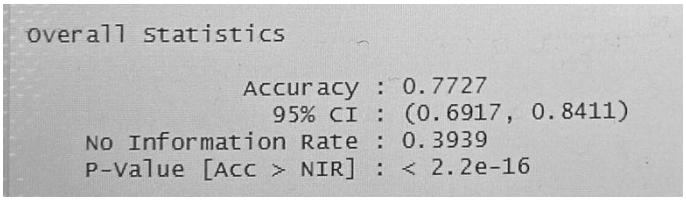

Figure 2

This model took a long time to train on our computer (12+ hours for one developer). We wondered if we could decrease the time for the training without losing any accuracy. In order to decrease the computation time, we tried to use less variables in our second Random Forest model. We only used the top 20 attributes selected by the LVQ feature selection method. Using 80% of data for training and 20% for testing. We also used 10-fold cross-validation. By tuning the number of "tuneLength". We get several results. The accuracy rate is 69% when using tuneLength=19, 71.21% when using tuneLength=50, 72.23% when using tuneLength= 100 and 72.23% when using tuneLength=1000. Each of the models took less than 5 minutes to train. As we can see, increasing the tuneLength results in higher accuracy, however, there is a limit when using tuneLength=100. We successfully shortened the compute time to only 5 minutes, however we sacrificed accuracy (accuracy rate decreased by 5.04%). In order to avoid any loss on the accuracy, we implemented a third Random Forest model.

In the third Random Forest model, we added more attributes in the model. With the top 31 attributes selected by LVQ, splitting 80%/20% for training and testing set and using 10-fold cross-validation, we have 78.03% accuracy in this model when using tuneLength=100, as shown in *Figure 3*.

```
               Accuracy : 0.7803
                 95% CI : (0.7, 0.8477)
    No Information Rate : 0.3939
    P-Value [Acc > NIR] : < 2.2e-16
```

Figure 3

In the fourth Random Forest model, we tried to use a more advanced cleaning data method, MICE. With the same setup in the third model, we had 74.24% accuracy, as shown in *Figure 4*.

```
              Accuracy : 0.7424
                95% CI : (0.6591, 0.8146)
    No Information Rate : 0.3939
    P-Value [Acc > NIR] : 4.677e-16
```

Figure 4

K Nearest Neighbor:

We used data cleaned by mean value method. Using the top 31 attributes selected by LVQ method, splitting 80%/20% for training and testing set and using 10-fold cross-validation, we have 65.15% accuracy rate, shown in *Figure 5.*

```
              Accuracy : 0.6515
                95% CI : (0.5637, 0.7323)
    No Information Rate : 0.3939
    P-Value [Acc > NIR] : 2.021e-09
```

Figure 5

Naïve Bayes Classifier:

We used data cleaned by mean value method. Using the top 31 attributes selected by LVQ method, splitting 80%/20% for training and testing set and using 10-fold cross-validation, we have 65.91% accuracy rate, shown in *Figure 6.*

```
               Accuracy : 0.6591
                 95% CI : (0.5716, 0.7393)
    No Information Rate : 0.3939
    P-Value [Acc > NIR] : 6.833e-10
```

Figure 6

Comparison Table:

Model	Type	Cleaning Method	Feature Selection Method	Number of attributes	Model Tune Length	Compute Time	Accuracy
1	RF	Mean value	Embedded RF	366	365	12+ hours	77.27%
2	RF	Mean value	LVQ	20	100	5 mins	72.23%
3	RF	Mean value	LVQ	31	100	5 mins	78.03%
4	RF	MICE	LVQ	31	100	5 mins	74.24%
5	KNN	Mean value	LVQ	31	default	<1 mins	65.15%
6	NBC	Mean value	LVQ	31	default	<1 mins	65.91%

Table 1

Conclusion:

In conclusion, indeed Random Forest with the LVQ feature selection, increased tune-length and good data cleaning techniques was our best model, providing an accuracy of 78.03%. This was better than the other random forest models using MICE cleaning method, embedded feature selection and LVQ feature selection method for the most important 20 features. The higher tune-length also helped because it allowed the system to tune the algorithm automatically. For example, our tune length of 500 made a developer's lower compute processing laptop freeze, such that the highest tune length that ran to completion successfully was k=365, meaning k

different values were assayed to find the optimal value based on these k values in the data. Further refactoring of the code however could also prove beneficial here.

Indeed, all Random Forest models exceeded Naïve Bayes & KNN. Naïve Bayes performed less well in part because, although it is robust to isolated noise points and irrelevant attributes, it relies on feature independence. [8] Indeed, for Naïve Bayes (our fifth model), from analyzing the columns, one sees that there are many highly correlated dependencies such as number of page visits and revisits for instance, as you can only revisit a page if you had already visited the page once. Naïve Bayes was too simplistic in its assumptions for this high dimensionality & low record count data, thus resulting in a worse accuracy value (65.91%). Our sixth model, KNN, determined the class label of a record by calculating the Euclidean distance to its nearest neighbors. The model however didn't perform as well for a few different reasons. For example, KNN doesn't perform well with high dimensionality data. [9] To remedy this one would need to have $n = k * 10^d$ to counter high dimensionality (d), so for d>10, too many data points would be needed to make the KNN an effective algorithm for use. [10] The data provided was constrained to the US region, and thus only had 664 respondents, and due to sparsity and high dimensionality, rendered KNN less effective than Random Forest.

There were some limitations in the study and areas for further study. Firstly, compute power posed an issue. The computer a developer in the team may have only had 4 cores, but had one been using 8+ cores it would have run faster and allowed more tests to be conducted, which would've prevented bottlenecking experimentation to re-compile higher tune length Random Forest models with compute intensive LVQ feature selection. Secondly, it would be good to allow the use of advanced cloud computing to further expedite the process if time and financial resources were not a problem as you could increase cores. For further study, one could also

investigate other helpful caret package functions like, preProcess, which conducts PCA variable selection, findCorrelation, which lets one obtain low correlational features by removing the ideal set of attributes, or nearZeroVar, which allows one to remove sparse variables. [11] Lastly, it would be helpful to make new variables and see if a new feature could be created (such as by adding or multiplying other features together) to yield a higher accuracy; indeed, adding smaller column values together could have added up to a fantastic predictive feature.

Appendix:

---

[1] Kang, H. (2013, May). The prevention and handling of the missing data. Korean journal of anesthesiology. Retrieved October 16, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3668100/

[2] Avcontentteam. (2020, July 5). R packages: Impute missing values in R. Analytics Vidhya. Retrieved October 17, 2022, from https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing values/#:~:text=MICE%20(Multivariate%20Imputation%20via%20Chained,of%20uncertainty%20in%20missing%20values.

[3] Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2020). 4.10.6 Random Forests. In Introduction to data mining (pp. 310–313). Pearson Education.

[4] Brownlee, J. (2020, August 14). Learning vector quantization for machine learning. Machine Learning Mastery. Retrieved October 16, 2022, from https://machinelearningmastery.com/learning-vector-quantization-for-machine-learning/

[5] Brownlee, J. (2019, August 22). Feature selection with the Caret R Package. Machine Learning Mastery. Retrieved October 19, 2022 from https://machinelearningmastery.com/feature-selection-with-the-caret-r-package/

[6] Clements, J. (2021, Feb 8). K-nearest neighbors (k-nn) explained. Medium. Retrieved October 19, 2022, from https://towarddatascience.com/k-nearest-neighbors-k-nn-explained-8959f97a8632

[7] Bhalla, D. (n.d.). Caret package implementation in R. ListenData. Retrieved October 19, 2022, from http://www.listendata.com/2015/03/caret-package-implementation-in-r.html

[8] He, Q. (2022, October). Session4 Alternative techniques in classification. Georgetown University, MATH 656. Washington, DC; Washington, DC.

[9] Damle, A. (2022, October). Lecture 2: k-nearest neighbors. CS 4/5780: Intro to Machine Learning. Retrieved October 15, 2022, from https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html.

[10] Damle, A. (2022, October). Lecture 2: k-nearest neighbors. CS 4/5780: Intro to Machine Learning. Retrieved October 15, 2022, from https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html.

[11] Caret (version 6.0-93). RDocumentation. (n.d.). Retrieved October 15, 2022, from https://www.rdocumentation.org/packages/caret/versions/6.0-93