IAES International Journal of Robotics and Automation (IJRA)
Vol. 10, No. 2, June 2021, pp. 123~132
ISSN: 2722-2586, DOI: 10.11591/ijra.v10i2.pp123-132

Missing data handling for machine learning models

Karim H. Erian, Pedro H. Regalado, James M. Conrad
ECE Department, University of North Carolina at Charlotte, Charlotte NC, United States

Article Info

Article history:

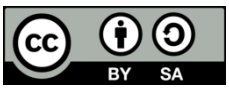
Received Jun 19, 2020
Revised Dec 1, 2020
Accepted Feb 12, 2021

Keywords:

Classification
Data pre-processing
Feature selection
Logistic regression
Machine learning
Target
Weights Fifth

ABSTRACT

This paper discusses a novel algorithm for solving a missing data problem in the machine learning pre-processing stage. A model built to help lenders evaluate home loans based on numerous factors by learning from available user data, is adopted in this paper as an example. If one of the factors is missing for a person in the dataset, the currently used methods delete the whole entry therefore reducing the size of the dataset and affecting the machine learning model accuracy. The novel algorithm aims to avoid losing entries for missing factors by breaking the dataset into multiple subsets, building a different machine learning model for each subset, then combining the models into one machine learning model. In this manner, the model makes use of all available data and only neglects the missing values. Overall, the new algorithm improved the prediction accuracy by 5% from 93% accuracy to 98% in the home loan example.



Corresponding Author:

Karim H. Erian
ECE Department
University of North Carolina at Charlotte
9201 University City Blvd, Charlotte, NC 28223, United States
Email: kerian@uncc.edu

1. INTRODUCTION

In supervised machine learning models, the accuracy of the model depends on the quality of the available dataset used for training. In most datasets, there are missing items. The way of handling the missing items is critical to train the model toward getting the expected results. The most common ways to handle missing items are by removal of the whole entry, by replacing the missing items by its default value, by changing the missing items to a zero, or by writing the average value instead.

In this paper, the team used an example of home loans dataset which includes 220 fields and by removing the missing items, the dataset shrunk from 307,511 entries to 8,603 entries and a 97.2% reduction in data size affected the performance of the estimation. The other solutions manipulate the data by introducing another value that is not the real value for this entry, leading also to lower accuracy [1]. This huge reduction urged the team to find another way to handle the missing entries. The team decided to come up with a way to pay attention to every unique cell that is blank, instead of just deleting a whole row if one cell is missing data. Especially when the team was aware that there were 220 unique features in the dataset, and how hard it was to gather more data.

The novel algorithm proposed in this paper aims to only remove the missing cells. This can be done by breaking the dataset into multiple subsets, each consisting of only one feature. Each subset is used to train a simple machine learning model. Then using a weighting mechanism, all models are combined into one model that uses all the available features.

2. THE PROPOSED ALGORITHM AND A SURVEY OF EXISTING METHODS

When researching ways to handle missing data, the team found that others have tried a variety of different methods, but only the top five methods for handling missing data will be discussed [2-7]. In each method, advantages and disadvantages are stated. The five methods found in research are:

- deleting rows,
- replacing with mean or median or mode,
- assigning a unique category,
- predicting the missing values, and
- using algorithms that support missing values.

2.1. Deleting rows

In the first method, the advantages of deleting rows can result in a robust and accurate model. Disadvantages of deleting rows is the loss of information and data. The biggest challenge is that if the missing values are a big percentage of the data, then the dataset is bound to be skewed.

2.2. Replacing with mean or median or mode

In the second method, advantages of replacing with the mean, median, or mode is that it can add variance to the dataset. This can be good when the dataset is small, and it also prevents data loss or removal. Disadvantages of replacing with the mean, median, or mode is that imputing the variations add variance and bias. Also, there are more complex imputation methods that are better, yet more complex.

2.3. Assigning a unique category

In the third method, advantages of assigning a unique category are that there are less possibilities with adding one extra category, this results in low variance. Disadvantages of assigning a unique category is that it adds less variance and also that by adding another feature to the model while encoding may have negative results in the original data and in return give a poor accuracy and performance.

2.4. Predicting the missing values

In the fourth method, advantages of predicting the missing values are that imputing the missing variable can be an improvement as long as the bias from the same is smaller than the omitted variable bias. Disadvantages of predicting missing values is that it is only truly considered a substitute for the true missing values. Also, the bias can also increase when an incomplete conditioning set is used for categorical values.

2.5. Using algorithms that support missing values

In the fifth method, advantages of using algorithms that support missing values is that they do not require creation of a predictive model for each cell with missing data in the dataset. Also, it negates the loss of data by adding a unique category. Disadvantages of using algorithms that support missing values is that it adds less variance and adds another feature to the model while encoding, and therefore may challenge the performance and accuracy.

2.6. Survey of related work

In the 'Background' section, there were five different methods discussed that were researched for different ways to handle missing data. Doing more research, the team found actual surveys of where some other researchers have used one of the five methods discussed previously. The top two surveys will be discussed [8-10]. In the first survey, there are two methods compared when using to help educational research. The two methods that were compared were stepwise regression which is the same as ignoring the missing data versus multiple imputation. Throughout the article, it talks about how in regression modeling the stepwise regression can be dangerous in missing files as the loss of information can be critical and the analyst may not even notice it. This is compared to one of the five methods researched in the sections above. It essentially removes the whole entry for any blanks found in the data and the article agrees that it can have potential loss of substantial amounts of information while also increasing the bias for the models applied. Then, the article discusses how multiple imputation is not yet being accepted by everyone because it involves adding simulated data to a raw data set and people think that there is a manipulation in the data and therefore not a good representation of the original data. Yet, imputation does the opposite as it tries to use the information available to simulate the missing data to minimize the bias.

The second survey reviewed the challenges of missing data in applying it to medical research. The article further discussed an imputation method stating that many people were not considering it and instead were considering case-wise deletion method. The data was a classification dataset and it helped to validate

and confirm one of the five methods discussed above in saying that the imputation method is not good enough for handling the missing data. It discusses handling missing data at random versus not at random stating the problems with alternative to data manipulation.

2.6. Proposed approach

The proposed approach to solve the problem of missing data is to take into consideration the unique value of a cell directly, instead of just deleting the whole entry. Instead of deleting the whole entry when finding a blank field, the team only deleted that one specific cell. The team combined multiple 1-feature models to create a new model. In this manner, the model created is first trained by each feature alone, then weights were combined together. In the 'Methods' section, the proposed approach is furthered discussed with the training of the 220 different models and features. Having all different weights from the different models, they will be combined together to have one big model without jeopardizing the integrity of the full missing entries, yet still being able to gather more accurate data for the features.

3. RESEARCH METHODS AND IMPLEMENTATION

3.1. Pre-processing the data

In this phase, the team had to understand the available data, how it is linked together, and how to find a target for any entry in the CSV files. It required a lot of intensive reading and researching previous implementations to see what tests others have implemented and why it worked or did not work at a certain prediction accuracy [11-13]. Then, it was noticed that the data included lots of string datatype variables. For example, 'Does the applicant have a car?' It would either be a 'yes' string variable or a 'no' string variable. String datatypes can be very complex to process, so a series of data dictionaries were implemented to map all values from categorical values to numerical values. For the converted cells, if the cell was empty a value was assigned to an integer variable of '99999999'. Also, the target value was mapped to an integer variable of 1 identifying that it was fine to lend the applicant money and an integer variable of 0 to identify a high risk applicant. Then, the logistic regression model was trained on 80% of the data and tested using 20% of the data. The results gave an acceptable accuracy percentage of 91%. Figure 1 is a sample of the dictionaries implemented.

```
In [52]: #dictionary:
         Contract_dict = {'Cash loans' :0, 'Revolving loans' :1}
         Gender_dic = {'M' :0, 'F': 1,'XNA':2}
         Own_Car_dic = {'Y':1,'N':0}
         Own_Realty_dic = {'Y':1,'N':0}

         NAME_TYPE_SUITE_dict = {'Unaccompanied':0,'Family':1,"Spouse, partner":2,'Children':3,'Other_A':4,'Other_B':5,'Group of
         NAME_INCOME_TYPE_dict = {'Working':0,'State servant':1,'Commercial associate':2,'Pensioner':3,'Unemployed':4,'Student':
         NAME_EDUCATION_TYPE_dict = {'Secondary / secondary special':1,'Higher education':4,'Incomplete higher':2,'Lower seconda
         NAME_FAMILY_STATUS_dict = {'Single / not married':0,'Married':1,'Civil marriage':2,'Widow':3,'Separated':4,'Unknown':5,
         NAME_HOUSING_TYPE_dict = {'House / apartment':0,'Rented apartment':1,'With parents':2,'Municipal apartment':3,'Office a
         OCCUPATION_TYPE_dict = {'Laborers':0,'Core staff':1,'Accountants':2,'Managers':3,'Drivers':4,'Sales staff':5,'Cleaning
         WEEKDAY_APPR_PROCESS_START_dict = {'WEDNESDAY':0,'MONDAY':1,'THURSDAY':2,'SUNDAY':3,'SATURDAY':4,'FRIDAY':5,'TUESDAY':6
         ORGANIZATION_TYPE_dict = {'Business Entity Type 3':0,'School':1,'Government':2,'Religion':3,'Other':4,'XNA':5,'Electric
         FONDKAPREMONT_MODE_dict = {'reg oper account':0,'org spec account':1,'reg oper spec account':2,'not specified':3,'':999
         HOUSETYPE_MODE_dict = {'block of flats':0,'terraced house':1,'specific housing':2,'':99999999}
         WALLSMATERIAL_MODE_dict = {"Stone, brick":0,'Block':1,'Panel':2,'Mixed':3,'Wooden':4,'Others':5,'Monolithic':6,'':99999
         EMERGENCYSTATE_MODE_dict = {'No':0,'Yes':1,'':99999999}
```

Figure 1. Sample dictionaries created

3.2. Understanding each feature impact on target

In order to understand each feature impact on the target value (which flowchart is shown by Figure 2), the team had to calculate the accuracy of each feature alone. This was done as part of data understanding. It was also done at first when trying to choose specific features, not to use all of them. In this case, the team implemented 220 models for all features, calculated the percentage and saved them. However, the first 120 features accuracies are represented in Figure 3, it was noticed that some of the features are have a very minimal impact of 7% prediction while others had maximum impacts of up to 90% accuracy. The team also noticed a lot of blanks were present in some features.

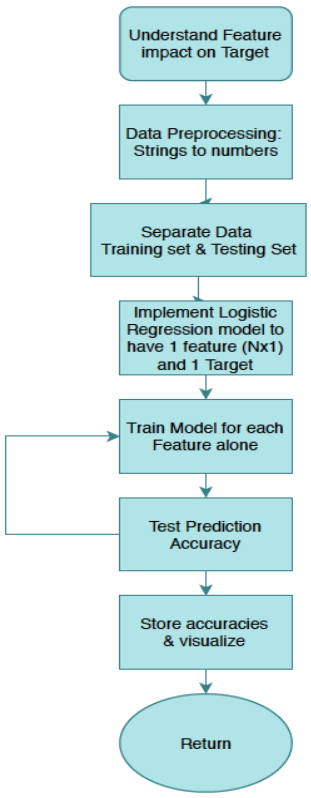


Figure 2. Features impact on target flowchart

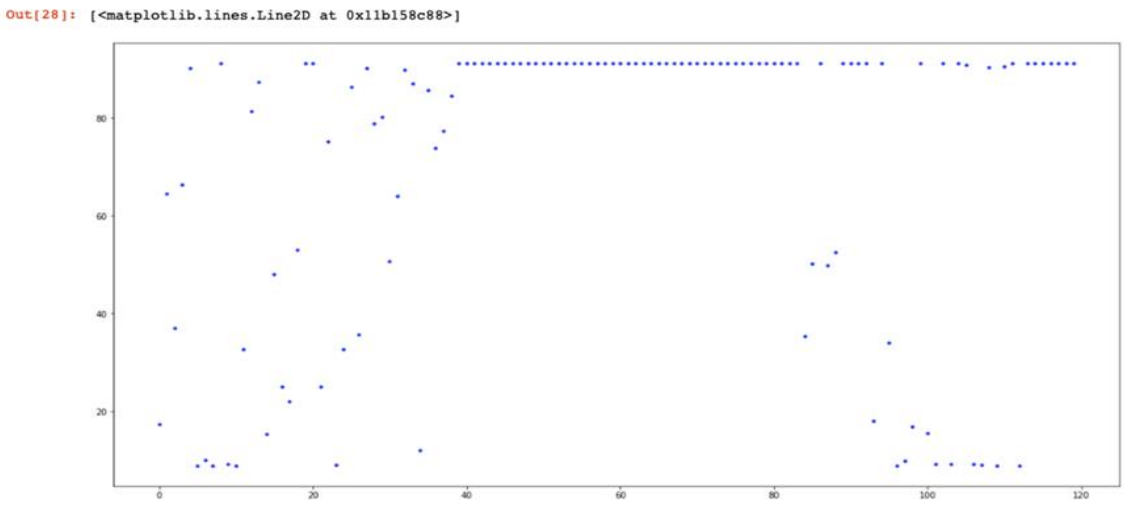


Figure 3. Image representing each feature impact on target prediction using logistic regression

3.3. Removing blanks and checking the accuracy

The team decided to prepare the data more to have it in a better shape and better usage. Thus, to remove blanks and start comparing features for the feature selection part. The team found that blanks are handled mainly in one of the 3 methods. Either by removing entries, or by filling in the blanks by default values or zeros. The team tried removing the whole entries, but that cause the data size to shrink from 307,511 entries down to 8,603 entries. Also, the prediction accuracy became much lower and the feature effect on the target prediction decreased significantly. This was mainly the main motivation to switch the focus on how to handle missing data. Figure 4 illustrates the impact of each feature on target prediction, while Figures 5-6 shows the process flow.

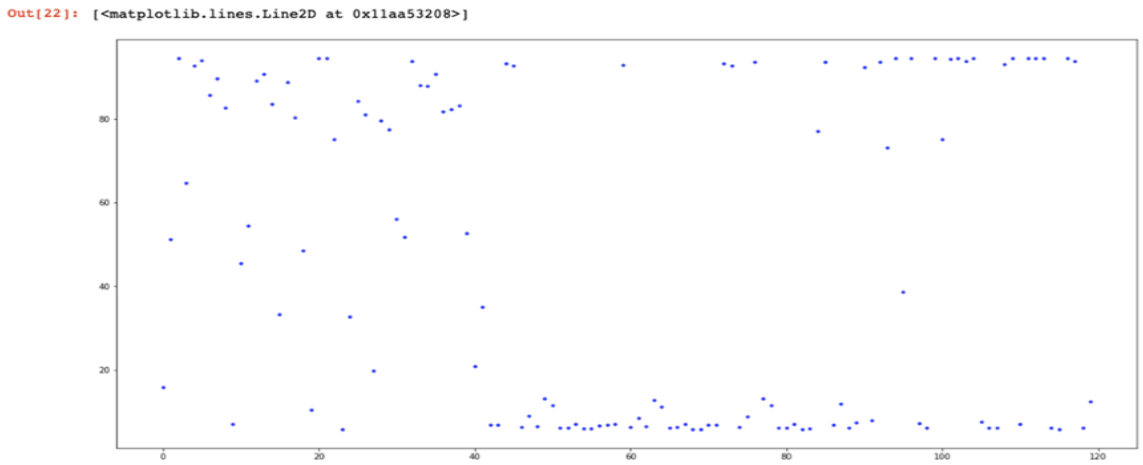


Figure 4. Feature impact on target prediction using logistic regression after removing entries with missing data

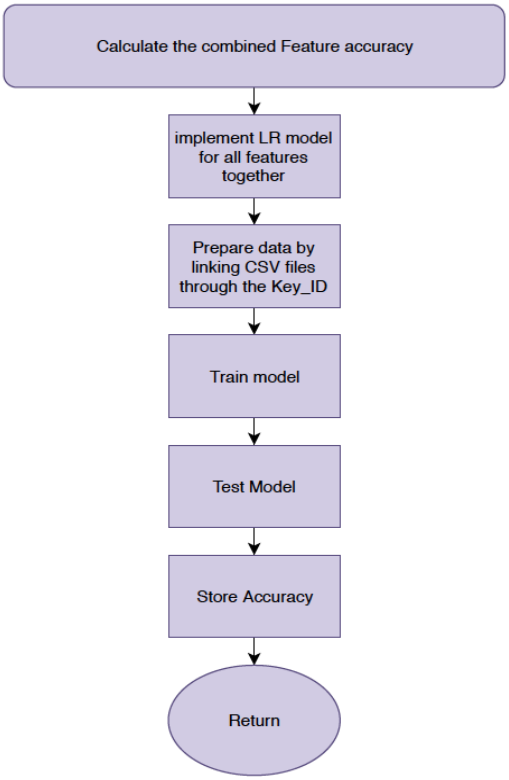


Figure 5. Combined features accuracy calculation flowchart

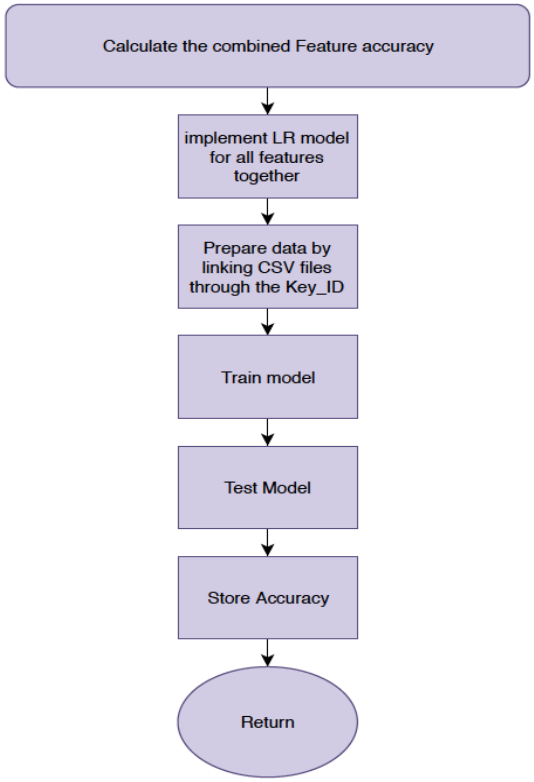


Figure 6. Blanks special treatment flowchart

3.4. Handling missing data

The implemented approach is about handling missing data in a new machine learning model. After removing all entries with missing data, the accuracy declined enormously. For this reason, the team decided to have another model that is a combination of multiple 1-feature models. In each model, the team handled the missing data in one of the already known methods which included applying the feature's default value, getting the average if the feature is a numerical data, or just removing the missing cells. The model was applied to each feature alone in order to find the best approach for each one. The next step the team took was to train 220 different models for the 220 features. Then, having the different weights from the different models, the team combined them together to have one model that is not ruined by missing data and at the same time, getting accurate data for all features. However, this was not enough because if the feature used alone to train the data is hardly related to the target, it may affect the overall model [8] (see Figure 7). So further mathematical solutions were applied (as illustrated by Figure 8):

- Multiply the new weights by old weights from the model used to train the 220 features together.
- Multiply the new weights by the testing accuracy from each separated model before removing blanks.

$$y = w_0.x_0 + b_0 = w_1.+x_1 + b_1 \ = w_2.x_2 + b_2$$
$$= w_3.x_3 + b_3 = w_4.+x_4 + b_4 = w_5.x_5 + b_5$$
$$= w_6.x_6 + b_6 = w_7.+x_7 + b_7$$
$$\text{if: } w = [w_0 w_1 w_2 w_3 w_4 w_5 w_6 w_7]$$
$$\text{and: } b = [b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7]$$
then
$$Y = w.X + b$$
$$= w_0.x_0 + b_0 + w_1.+x_1 + b_1 + w_2.x_2 + b_2$$
$$+ w_3.x_3 + b_3 + w_4.+x_4 + b_4 + w_5.x_5 + b_5$$
$$+ w_6.x_6 + b_6 + w_7.+x_7 + b_7$$

But this is equal to 8T not T.

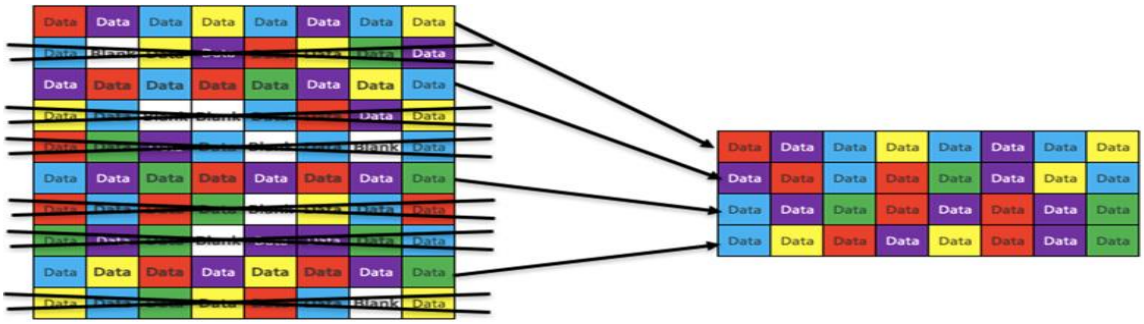


Figure 7. Old method of handling missing data, all white squares are presenting missing data

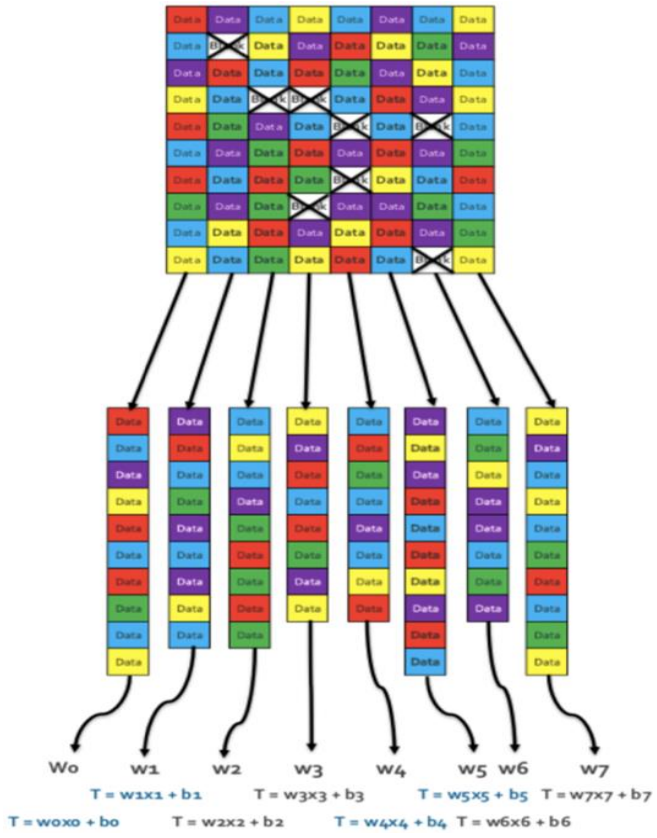


Figure 8. Proposed method of handling missing data (white squares = missing data)

a. Method I: Divide it by the number of features. In this example, it should be divided by 8 features, but in the project, it was divided by 220 features.

$$Y = w.X + b = (w_0x_0 + b_0 + w_1x_1 + b_1 + w_2x_2 + b_2 + w_3x_3 + b_3 + w_4x_4 + b_4 + w_5x_5 + b_5 + w_6x_6 + b_6 + w_7x_7 + b_7)/8$$
$$So\ w = [w_0\ w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7]/8$$
$$\text{and: } b = [b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7]/8$$

b. Method II: When linear regression was first used, the weight vector was derived, (assume it as 'v' instead of 'w' not to conflict it with the 'w' that has already been discussed).

Let v = [v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7]. Since it is known that the weight sum is always equal to 1, then an equation can be established such that $v_0 + v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7 = 1$. In this case, each element of w was multiplied by the corresponding element of v. The higher impact weight in the w vector is enhanced by the weight from the initial linear regression weights v. Therefore, w = [$w_0.v_0$,$w_1.v_1$,$w_2.v_2$,$w_3.v_3$,$w_4.v_4$,$w_5.v_5$,$w_6.v_6$,$w_7.v_7$].

3.4. Creating 220 models

To implement this approach, the team needed to build a machine learning model for each feature alone. The team decided to use the linear regression model for two main reasons. The first main reason that a linear regression model was used was because it was cleaner to build the weighting vector from the separate weights [14]. The second reason that a linear regression model was used was because it had a higher accuracy in normal cases for this dataset than the logistic regression model which compared 93% vs 91%, respectively.

To implement the models, the team set up the 'X' matrix first. Then, 220 'X' vectors were built in each one. Thereafter, all the blank data were removed from all the gathered CSV files. After this setup, the team continued to gather the corresponding target vector for each input data vector. This was done because when the blank data is removed from only one column (not all columns together), the 'T' vector was needed to match the data input. In this process, the challenge was that not all CSV files had the target column.

Although, it was only the "application_train.csv" file, the rest of the files had an attribute that was called "Key ID Current" that was present in some of the CSV files including in the "application train.csv" file. The team used the IDs in these CSV files to be able to link the target to the input data from the other CSV files. Then, it was noticed that some of the CSV files did not have the "Key ID Current" attribute present, but there was another ID that existed that could be used with the rest of the files because it linked up each of those IDs nicely for all of the files. In this manner, the team used the secondary ID found to get the "T" vector for all input data [15].

3.4. Creating new model

Now, having the 220 'X' vectors and the 220 'T' vectors, the team divided them into 80% training data and 20% testing data. The team trained all models and got 220 'w' and 220 bias values. Method I and Method II were implemented to combine and create the 'w' vector for the new model. Figure 9 is a flowchart describing the new approach. Color light orange represents the novel idea. The main flow chart contains rectangles and boxes. The boxes are explained in more detail in other following flowcharts with similar color of the original box.

4. RESULTS AND DISCUSSION

4.1. Creating testing dataset

For testing, the team could not just use each file separately like it was done for the training part. Therefore, the team mapped all the data from all files using the IDs found in the CSV files. This step took way more time than expected. The team searched for a method to do this process of mapping without 'for' loops to reduce time complexity, especially with the large dataset that was available, but unfortunately, there was no easy method found. The method that was used to combine the data is the following: First, the team initialized a new 'X' array as a set of zeros array with the size of 307511 rows and 220 columns. Then, using the "application train.csv" file as the main file, the team looped over all IDs in the file and searched for the matched ID in the "Credit _Card Balance.csv" file. In the case that a match was found, the team appended the values from the "Credit Card Balance.csv" file to the 'X' array which already had the values from the "application train.csv" file. The team repeated this process for all the CSV files so that the 'X' matrix was available. The data was then separated into the training set and testing set, such that the testing dataset was not used before in the training [16].

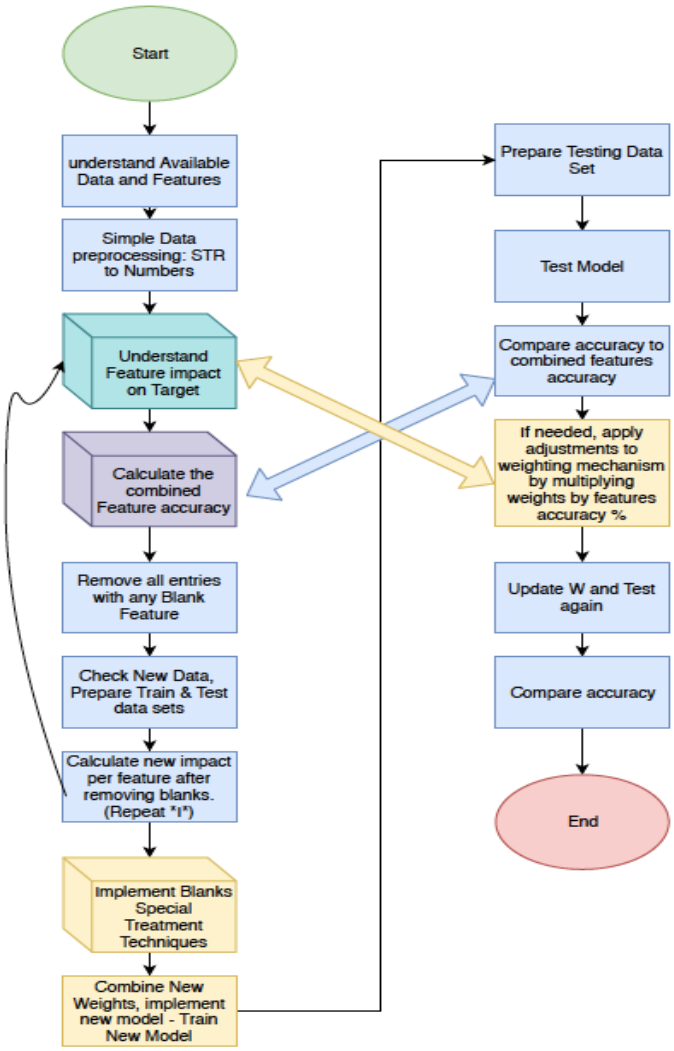


Figure 9. Approach main flowchart

4.2. Results for testing the new model

Having the testing dataset established and the '*w*' ready for the two methods, testing took place. It was found that the accuracy for both methods increased to 98%. This is a 5% increase in prediction accuracy from the old methodology which had an original accuracy of 93%. The team had to review to find out why both methods resulted in 98% prediction accuracy. The team found that the difference in the two methods did not affect the final result by a large margin because '*Y*' is normalized at the end. If the value ended up being greater than 0.5, then it was rounded up to 1, and if the value was less than 0.5, then it was rounded down to 0.

4.3. Discussion

4.3.1. Application

This research has been done in parallel to another research about autonomous control of an all-terrain vehicle (ATV) in the woods. The research was divided into phases. The first phase [17-19] was to control the ATV using algorithms. The second phase was to gather data from the ATV while moving [20] and use the data to implement a rating mechanism to identify different driving behaviors. The third phase is to use the data gathered with the method invented in this paper in order to control the ATV using machine learning techniques.

4.3.2. Lessons learned

Through this project, the team was able to learn a variety of technical and soft skills. Some of the technical skills learned through this project were the exploration of the linear regression model in comparison

to the logistic regression model to see which would best fit the data of this specific classification project. At first, one of the most challenging things for the team was choosing a project which contained enough data. In machine learning, the team learned that it is very important to have as much raw data as possible. This was essential for training and being able to generalize better for testing whichever model chosen. Another skill learned through this research project is that it is very important to pre-process the data carefully and thoroughly. If the data is not pre-processed properly, it will be trained in the wrong manner and therefore give skewed results. Therefore, pre-processing the data is as important if not more important than training it and testing it.

Other skills that were learned throughout this project were the continuous implementation of programming in Python language and using its libraries to make use of all the resources available to analyze the data. This was a learning curve, but it helped in the long run in being able to handle data from multiple CSV files at the same time. This along with handling Python's classes, methods, and variables between multiple Jupyter Notebooks and multiple machines were things that had to be learned in order for the team not to waste time doing things manually or one step at a time. Other minor lessons were learned such that it helped reduce the file size and computation time. Examples included things such as removing 'print' commands if they were not needed, reducing loops, reducing the number of cells in a notebook, and having separate project classes into multiple notebooks. All of this made the pre-processing, training, and testing part of the project much more bearable keeping time complexity in mind.

Overall, it was learned that by using linear regression model and taking each blank cell as a unique attribute to handling missing data, the data accuracy was able to be enhanced by 5% from 93% to 98%. Future improvements will include the use of more complex models such as using neural networks to see if the data accuracy can increase to be even slightly better or if there are any changes as well as challenges that come with it. Other future improvement included how to implement missing data handling techniques in other more complicated models such as reinforcement learning using greedy algorithms to test the agent learning and different methods in unsupervised learning methods such as k-nearest neighbors (KNN) to show potential discrete and continuous attributes [14].

5. CONCLUSION

Using the conventional methods for handling missing data, the data were reduced to 8,603 entries instead of 307,511 entries of the original dataset. The novel algorithm discussed in this paper allows the usage of all the available data. The new method enhanced the accuracy of the machine learning model tested from 93% using the conventional algorithms to 98% using our novel algorithm. The novel method extends further than just this research as it can be applied in different areas including training networks like the ATV mentioned in the discussion section above.

ACKNOWLEDGEMENTS

Thanks to Dr. Minwoo (Jack) Lee for his guidance in this research.

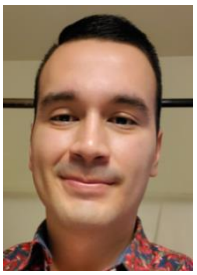
REFERENCES

[1] B. Angelov, "Working with Missing Data in Machine Learning," in *Towards Data Science*, 2017. [Online]. Available: https://towardsdatascience.com/working-with-missing-data-in-machine-learning-9c0a430df4ce.
[2] K. Maladkar, "5 Ways to Handle Missing Values in Machine Learning Datasets," 2018, Available: https://www.analyticsindiamag.com/5-ways-handle-missing-values-machine-learning-datasets/.
[3] M. Pampaka, G. Hutcheson, J. Williams, "Handling missing data: analysis of a challenging data set using multiple imputation," *International Journal of Research & Method in Education*, vol. 39, no. 1, pp. 19-37, 2016.
[4] K. L. Masconi, T. E. Matsha, J. B. Echouffo-Tcheugui, R. T. Erasmus, A. P. Kengne, "Reporting and handling of missing data in predictive research for prevalent undiagnosed type 2 diabetes mellitus: A systematic review," *EPMA Journal*, vol. 6, no. 7, pp. 1-11, 2015.
[5] Alijs, "Home Credit Default Risk Kaggle competition 3rd place solution," *Kaggle*, 2018. [Online]. Available: https://www.kaggle.com/c/home-credit-default-risk/discussion/64830.
[6] Maxwell, "Home Credit Default Risk Kaggle Competition 2nd place solution (team ikiri DS)," in *Kaggle*, 2018. [Online]. Available: https://www.kaggle.com/c/home-credit-default-risk/discussion/64510.
[7] B. Tunguz, "Home Credit Default Risk Kaggle Competition 1st Place Solution," in *Kaggle*, 2018. [Online]. Available: https://www.kaggle.com/c/home-credit-default-risk/discussion/64821.
[8] A. Swalin, "How to Handle Missing Data," in *Towards Data Science*, pp. 1-10, 2018.
[9] Kaggle "Home Credit Default Risk-Kaggle," 2018, [Online] Available: https://www.kaggle.com/c/home-credit-default-risk/data.

[10] W. Koehrsen, "Start here: A gentle introduction," in *Kaggle*, 2018. [Online] Available: https://www.kaggle.com/willkoehrsen/start-here-a-gentleintroduction/notebook.
[11] K. H. Erian, S. Mhapankar, J. M. Conrad, and S. Gambill, "System Integration over a CAN Bus for a Self-Controlled, Low-Cost Autonomous All-terrain Vehicle," *2019 SoutheastCon*, Huntsville, AL, USA, pp. 1-8, 2019, doi: 10.1109/SoutheastCon42311.2019.9020379.
[12] K. H. Erian, "System Integration Over a Can Bus for a Self-Controlled, Low-Cost Autonomous All-Terrain Vehicle," MSc thesis, The University of North Carolina at Charlotte, pp. 1-18, 2019.
[13] K. H. Erian, J. M. Conrad, "Measuring driving performance for an All-Terrain Vehicle in a paved road in the woods," *2020 SoutheastCon*, Raleigh, NC, USA, pp. 1-8, 2020, doi: 10.1109/SoutheastCon44009.2020.9249705.
[14] G. Hutcheson, "Missing data: Data replacement and imputation," *Journal of Modelling in Management*, vol. 7, no. 2, pp. 1-19, 2012.
[15] J. R. van Ginkel, M. Linting, R. C. Rippe, A. van der Voort, "Rebutting existing misconceptions about multiple imputation as a method for handling missing data," *Journal of Personality Assessment*, vol. 102, no. 3, pp. 297-308, 2020.
[16] C. M. Musil, C. B. Warner, P. K. Yobas, and S. L. Jones, "A comparison of imputation techniques for handling missing data," *Western Journal of Nursing Research*, vol. 24, no. 7, pp. 815-829, 2002.
[17] J. W. Graham, P. E. Cumsille, A. E. Shevock, "Methods for handling missing data," *Handbook of Psychology*, Second Edition 2, 2012.
[18] J. Gantz and D. Reinsel, "Extracting value from chaos," *IDC IVIEW Extracting Value from Chaos*, IDC. 1142, pp. 1-12, 2011.
[19] J. R. Cheema, "Some general guidelines for choosing missing data handling methods in educational research," *Journal of Modern Applied Statistical Methods*, vol. 13, no. 2, pp. 53-75, 2014.
[20] S. Mhapankar, "A Navigation System for Low-cost Autonomous All-terrain-vehicles," M.Sc. thesis, University of North Carolina at Charlotte, pp. 1-70. 2019.

BIOGRAPHIES OF AUTHORS

Karim H. Erian received his B.S. degree in Electronics and Communications Engineering from Cairo University, Egypt, in 2008 and his M.S. degree in Electrical Engineering from the University of North Carolina at Charlotte in 2019. He is currently pursuing his Ph.D. degree in Electrical Engineering. From 2009 to 2013 he was an embedded software engineer at Valeo. From 2013 to 2017 he was a senior principal quotation engineer at Valeo. From 2017 to 2019 he was a Fulbright scholar at UNCC. He worked as a teaching assistant from 2018 to present.



Pedro H. Regalado received his B.S and M.S. degree in Electrical Engineering from the University of North Carolina at Charlotte in 2018 and 2019, respectively. He is currently pursuing his Ph.D. degree in Electrical Engineering. His research topics include the Internet of Things, wearable technology, and intelligent sensors with a focus in mixed reality. From 2016 to 2020 he experienced a variety of internship roles from multiple respectable companies including Cisco Systems, The Boeing Company, Monster Smart Solutions, Rockwell Automation, and Alston & Bird as a network engineer, electrical design engineer, embedded systems engineer, technical sales analytics engineer, and a patent specialist, respectively. He worked as a teacher assistant from 2017 to present and is continuing his doctoral degree under the Graduate Assistant Support Program and NSF Ventureprise I-Corps sponsored funding.

James M. Conrad received his B.S. degree in Computer Science from the University of Illinois, Urbana, in 1984 and his M.S. and Ph.D. degrees in Computer Engineering from North Carolina State University in 1987 and 1992, respectively. From 1984 to 1990 he was a software engineer with IBM, from 1992 to 1995 he was an assistant professor at the University of Arkansas, and from 1997 to 2003 he was a software engineer and project manager with Ericsson and Sony Ericsson. Since 2003 he has been an associate professor and professor with the Electrical and Computer Engineering Department at the University of North Carolina at Charlotte. He is the author of eight books and more than 140 articles in the areas of embedded systems, robotics, parallel processing, and engineering education.