Final Project Report

Kel Gruber
University of Missouri-St. Louis

11/29/23

Contents

Abstract

The goal of this project is to build a neural network model that if given a banking client can predict if the client will make a term deposit at this bank or not. This project will explore the entire process of solving a Binary Classification problem beginning with cleaning and preparing the data set, creating, building, evaluating, and choosing the best neural network prediction model, and determining feature importance and feature reduction to build better models.

1 Introduction

Most people today have a bank account at some form of banking institution. The most common is the commercial bank which provides basic banking services and products to the general public. While most people are familiar with banking fees, commercial banks make money primarily by providing loans and collecting the earned interest on these loans. The bank relies on clients depositing funds into their bank accounts to provide funds to loan to other clients. However, in standard checking and savings accounts the account holder may withdraw funds at any time which makes it difficult for the bank to determine what loans it can finance at a given time. To address this problem banks offer term deposit accounts. Customers who open one of these term deposit accounts have an agreement with the bank that they will not withdraw their funds for a fixed time period. In exchange, the bank will pay the account holder a higher interest rate on these funds. This is an extremely safe investment for the account holder and typically term deposit accounts are appealing to low-risk investors. It is advantageous for a bank to promote term deposits to its clients because the more funds that are fixed the more loans or larger loans the bank can issue and earn interest from.

I selected this particular problem and dataset because it struck me as a common real-life problem. Finding clients and customers interested in making a term deposit is a common challenge that financial institutions face, and being able to make predictions on which customers might open a term deposit and those that will not, allows the bank to better target their campaign and resources to contacting interested customers. It also allows the bank to issue more loans overall which is important to the continued success of the bank and its services. I believe that finding an effective prediction model for this problem can have practical implications for marketing strategies, resource allocation, and overall customer relationship management.

2 Dataset

This project uses the "Bank Marketing Data Set" from Kaggle Data Science and uses data obtained from direct marketing campaigns of a Portuguese banking institution to its clients. These contacts were made by phone calls to either the customer's home telephone or cellphone.

This data set consists of 41,188 instances and 20 variables. These variables are a mix of continuous data and non-numeric classification data. The original output variable is labeled 'y' and has 36,548 'no' values and 4,640 'yes' values. For clarity, this column was immediately relabelled to output, and the values were converted to binary values where no is 0 and yes is 1. Input Feature Information:

1. age : (numeric) - age of the client
2. job : (categorical) - type of job
3. marital : (categorical) - marital status
4. education : (categorical) - education level
5. default: (categorical) - If the client has credit in default
6. housing: (categorical) - If client has a housing loan
7. loan: (categorical) - If client has personal loan

8. contact: (categorical) - Contact communication type
9. month: (categorical) - Last contact month of year
10. day.of.week: (categorical) - Last contact day of the week
11. duration: (numeric) - Last contact duration, in seconds
12. campaign: (numeric) - number of contacts performed during this campaign and for this client
13. pdays: (numeric) Number of days that passed by after the client was last contacted from a previous campaign; 999 means client was not previously contacted
14. previous: (numeric) number of contacts performed before this campaign and for this client
15. poutcome: (categorical) - Outcome of the previous marketing campaign
16. emp.var.rate: (numeric) - Employment variation rate - quarterly indicator
17. cons.price.idx: (numeric) - Consumer price index - monthly indicator
18. cons.conf.idx: (numeric) - Consumer confidence index - monthly indicator
19. euribor3m: (numeric) - The Euribor 3 month rate - daily indicator
20. nr.employed: (numeric) - Number of bank employees - quarterly indicator

3 Data Processing

Initially, there were 41,188 instances, 20 input variables, and 1 output variable in this dataset which is plenty of data for building a neural network model. The features all belong to one of three categories, the first is data about the bank customer and their current financial situation, the second is data about whether the bank has contacted the person before in prior marketing campaigns, and the third is data that is related to the current state of the economy and market in Portugal.

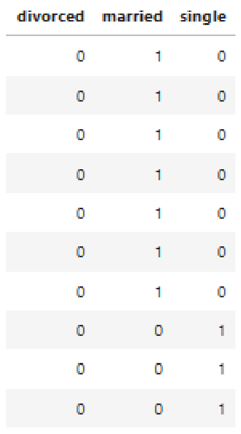
Upon loading the data I also reviewed the features themselves and removed the features that were not relevant to our output variable. For example, the contact column only contains two categories 'telephone' and 'cellphone'. These two categories are essentially the same form of communication so it will not impact whether a client opened a term account or not. So the contact column, number of employees employed at the bank columns, and the month and day of the week a client was contacted were removed. Columns that are all one value or do not directly relate to the problem do not affect the output variable but can create noise in the model and its predictions. I also decided to drop the duration column which stored the total seconds the bank spoke with a customer on the phone, per the author's suggestion that can lead to bias in the model and poor predictions.

Additional review and cleaning led to removing the default column because it had 8,597 'unknown' values, only 3 'yes' values, and the remainder were 'no' values. Since this column was almost entirely 'no' values it was removed to prevent adding noise to the model.

I decided for the remainder of the columns with 'unknown' values that columns since they were approximately only 2-3% of the total data that I could safely remove these rows from the dataset. In addition, since most of this data was categorical, attempting to replace the 'unknown' values with a mean value was not. So all the rows that had 'unknown' values in the education, job, marital status, housing, and loan columns were then removed.

Now that I had handled any missing values it was time to convert any categorical data from non-numerical data to numeric data. For the housing, loan, and output columns, this process was simple. I just replaced the 'no' values with 0 and the 'yes' values with 1. For the education column, there were 7 categories present: illiterate, basic 4y, basic 6y, basic 9y, high school, professional course, and university degree. This column has ordinal data meaning that there is a basic hierarchical structure to them. For example, being illiterate is the equivalent to no education while a university degree is the highest education an individual will receive. I converted these 7 categories to numbers that correspond to the height of the completed education. 'Illiterate' was replaced with 0, 'university degree' was replaced with 6 and the other values were ranked in order of education completion accordingly from 1 to 5.

The marital and job columns are categorical data with no hierarchical structure to them making them more difficult to convert to numeric data. For example, single, married, and divorced have no innate numerical structure so categorizing these variables with 1,2, and 3 could potentially introduce bias and error to our model. This is because the model might try to understand why the divorced value of 3 might be worth more than a single value of 1, or for the model to conclude that divorced (3) is equal to single (1) plus married (2). To solve this problem I used One-Hot Encoding to convert the marital column into 3 columns and the job column into 11 columns that use binary values instead of categorical numbers to group the data.



divorced	married	single
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	0	1
0	0	1
0	0	1

Figure 1: Depiction of the 3 new columns added to the dataset after using One-Hot Encoding to convert the 3 categories of single, married, and divorced to numeric data.

At the end of this process, I was left with a total of 26 input feature variables. Final Feature Information:

1. age - age of the client
2. job - admin.
3. job - blue-collar

4. job - entrepreneur
5. job - housemaid
6. job - management
7. job - retired
8. job - self-employed
9. job - services
10. job - student
11. job - technician
12. job - unemployed
13. marital - single
14. marital - married
15. marital - divorced
16. education Client education level
17. housing - If client has a housing loan
18. loan - If client has personal loan
19. campaign - number of contacts performed during this campaign and for this client
20. pdays - Number of days that passed by after the client was last contacted from a previous campaign; 999 means client was not previously contacted
21. previous - number of contacts performed before this campaign and for this client
22. poutcome - Outcome of the previous marketing campaign
23. emp.var.rate - Employment variation rate - quarterly indicator
24. cons.price.idx - Consumer price index - monthly indicator
25. cons.conf.idx - Consumer confidence index - monthly indicator
26. euribor3m - The Euribor 3 month rate - daily indicator
27. output

4 Data Visualization and Data Analysis

To begin visualizing and exploring our dataset I decided to start with the heat map below. Using this I was able to see that most of the features present in our data do not have a strong correlation with the output value.

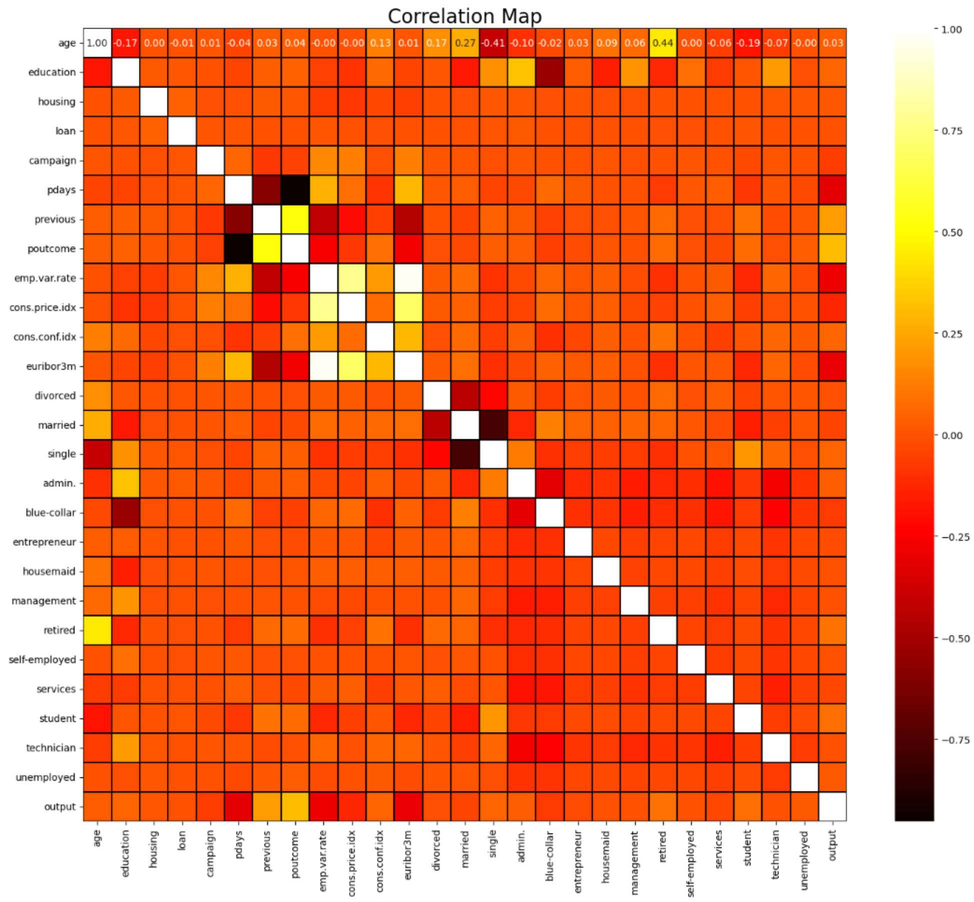


Figure 2: Heat map depicting the correlation between the 26 data features present in the dataset.

My initial hypothesis is that the best predictors of whether a person would open a term deposit account were features related to the person's financial status, such as if they have a loan or housing mortgage payment currently along with their economic class status such as their job and education level. For categorical and binary data histograms are not always the best for showing the distribution so I used pie charts for certain features to show the distribution of the categories. I also believe that for the One-Hot Encoded data like the marital status and job that the while the data needed to be encoded for the model that the initial piecharts showed the distribution and relations of the different categories better than the One-Hot Encoded pie charts so I have included those below. I am curious to see how the economic and market indicators such as the Employment variation rate, Consumer Price index, Consumer Confidence index and Euribor the Euro Interbank Offered Rate might have an impact on the clients investment decisions.

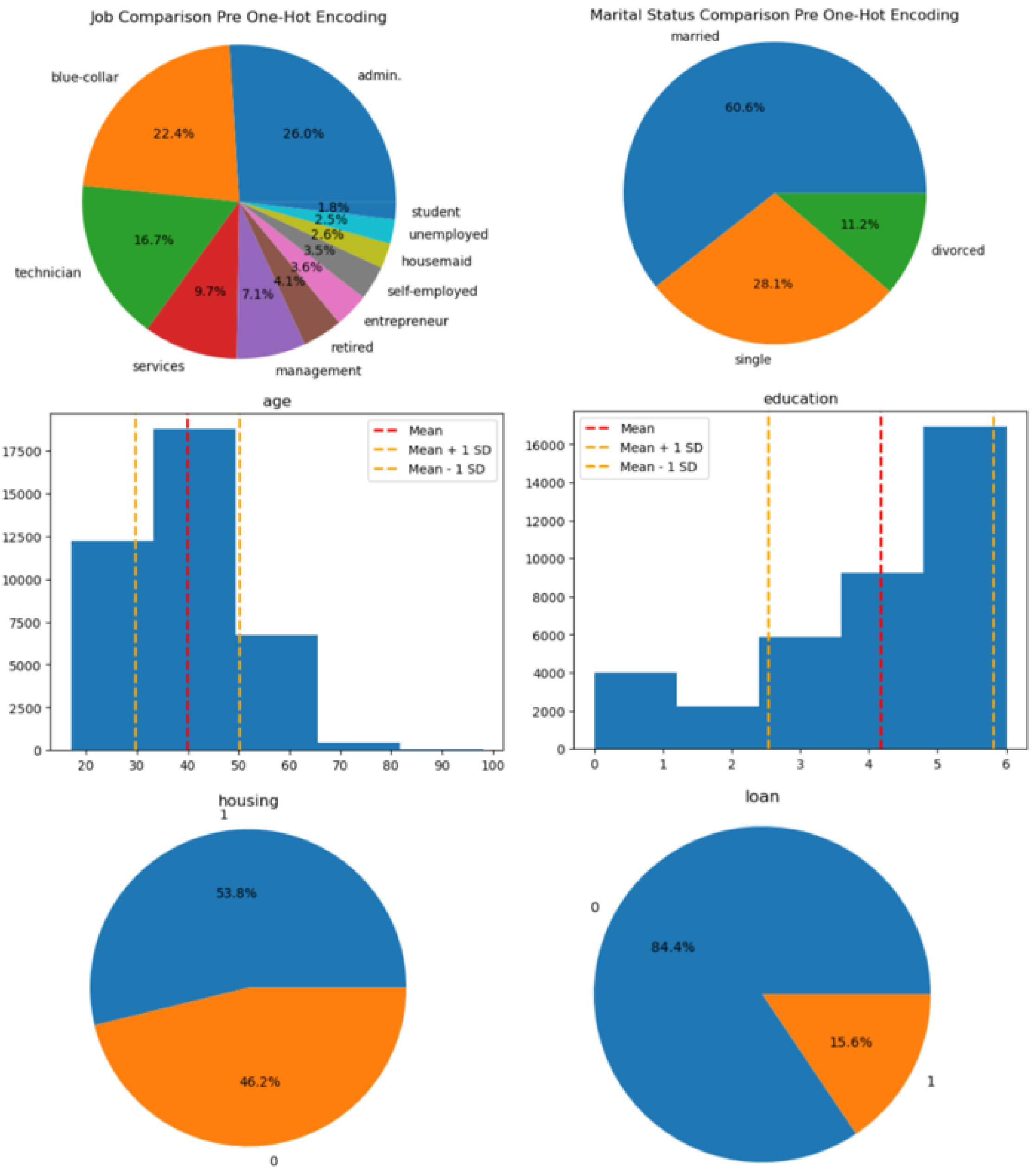


Figure 3:

5 Balancing Data

Initial data exploration showed that the original data was unbalanced and significantly favored the output '0' values so it was clear that before building models the data would need to be re-sampled. Balancing the data is important because an unbalanced dataset can lead to bias in the model in favor of the majority class. If the data is not re-balanced the model would be biased to predict that a client would not open a term deposit account.

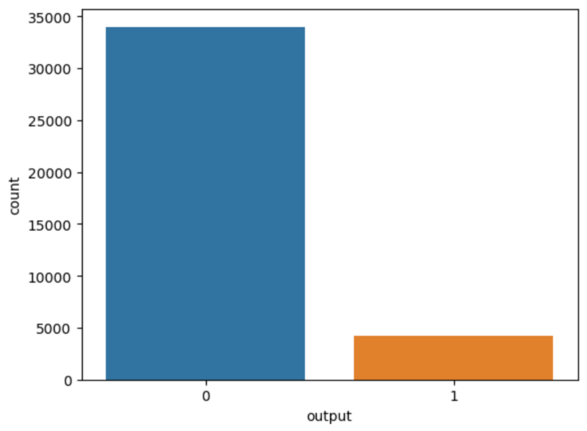


Figure 4: Depicts the significant imbalance in the output data after cleaning.

To prevent this bias I re-sampled the data by down-sampling the majority class. I reduced the total instances of the '0' output values to be equal to the total '1' output values. After re-sampling the dataset had a total of 8,516 instances and 26 features and an equal number of '0' and '1' output values. One downside to this method of data re-sampling is that while it was easy to implement I also removed a lot of data points from the dataset so it is possible that information present in those discarded instances could have helped the model make better generalizations.

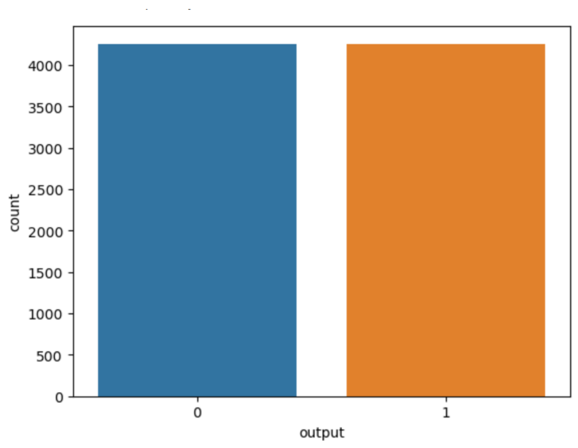


Figure 5: Output data distribution after re-sampling and balancing the Data with down-sampling.

6 Data Normalization

After splitting the data into training and testing sets, but before building any models I decided to normalize the data. Data normalization is when you re-scale the data values into the range between [0,1]. Neural network models tend to learn better when the data values are between 0 and 1. I have a lot of binary data values and features in this data set so normalization is not needed for those variables, but I do have some numeric data were greater value ranges such as the age, education, and economic feature columns. Normalizing these features should help produce a better model. To normalize the data I used the following Normalization equation:

$$x_{normalized} = \frac{(x - x_{minimum})}{(x_{maximum} - x_{minimum})} \tag{1}$$

7 Modeling

I decided to begin this phase of the project by building the simplest neural network model, the Logistic Regression model, with one neuron. It was logical to begin this phase of the project by building smaller models first to prevent overfitting or building unnecessarily large networks. I also tested various training schedules and total epochs with this first model and settled on training my models for 100 epochs. The early stopping flag was also turned on to prevent overfitting the model, and all models were set to stop training if no improvements were made to the val_loss metric after 20 epochs. Once I felt comfortable that I built the best logistic regression models I decided to try some different architectures to see how the size and shape of the model were related to the accuracy and other evaluation metrics.

In all, I built and tested 7 different neural network models. I also tried additional training for some of the models with lower accuracy values to see if I could improve the models with more training. Ultimately while the additional training did slightly improve accuracy in these models, the other evaluation metrics fell. I ultimately did not do additional training for most of the models because the early stopping flag had stopped the initial training early at 20-50 epochs so I did not feel that the additional training would improve these models and I wanted to avoid overfitting. Below are the results of these model experiments.

8 Evaluation

Model Type	Model Training Acc.	Validation Acc.	Precision	Recall	F1-Score
Logistic regression model	72.42%	71.64%	0.73	0.67	0.70
Neural network model (2-1)	73.39%	72.46%	0.77	0.63	0.69
Neural network model (4-1)	73.73%	72.05%	0.76	0.63	0.69
Neural network model (8-1)	73.40%	71.99%	0.77	0.62	0.68
Neural network model (8-4-1)	73.90%	72.17%	0.77	0.61	0.68
Neural network model (16-8-1)	73.90%	72.58%	0.78	0.62	0.69
Neural network model (32-16-8-1)	73.87%	72.40%	0.77	0.62	0.69

Table 1: Neural Network Models and Results.

At first glance, these models all appear to have similar results of about 71%-73% accuracy on the validation set. However, the additional evaluation metrics such as precision, recall, and f1 score set them apart. The Logistic Regression model surprisingly performed better in some ways than the larger architectures. It might have the lowest accuracy overall but it has the highest recall and F1 scores of all the models. The precision value for this model is close to the recall score which can be difficult because typically as precision increases the recall score drops. The Training and Validation Accuracy values are quite close together which shows that the model learned to make decent generalizations about the data during training so it performs similarly on the testing data.

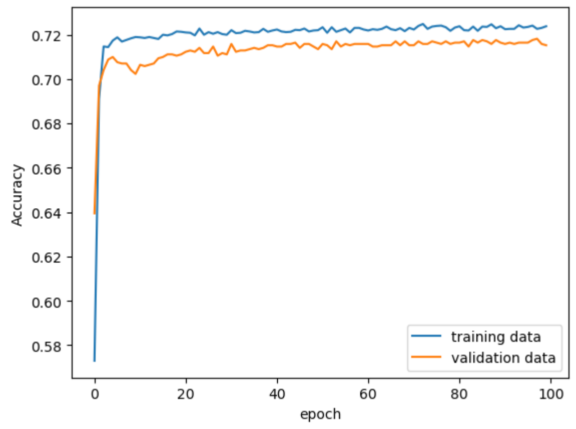


Figure 6: Depicts the training history for the logistic regression model.

The logistic regression model also has the highest recall and F1 scores of all the models, and the precision is close to the recall score as well. This indicates that the balance and completeness of the true positives predicted are good. Precision measures the ratio of correct positive predictions out of all positive predictions that were made. Recall is the measure of the total true positives divided by the total number of true positives and false negatives. The recall in our training data and testing data is quite different though, in the training data it is quite low indicating that there is a high number of false negatives. Typically the closer the higher precision gets the lower recall becomes which can be seen in some of the larger models that as the accuracy and precision creep up the recall and F1 score drop.

Finally, the f1 score combines precision and recall into one measure. This score is also best in the Logistic Regression model which is why I decided that this is the best model built overall because it is the most consistent in identifying the correct true positives. While the accuracy overall is the lowest I think in this problem identifying the people who are most likely to make term deposits is more useful to the bank marketing callers than correctly identifying everyone's final decision. In the other models, with the recall dropping along with F1-Score this means that the model is less likely to miss potential positives and gives the marketers a better list of customers they should contact about their term deposit accounts.

One important note on these results is that the other models are not technically worse than the Logistic Regression model, the precision values in those models are significantly better in those models but the drop in the recall and F1 scores is why I decided to not choose them. While the logistic regression model will return more values and potentially more false positives this model is for determining which customers are likely to open a term deposit account and will be used by the bank to determine which customers they should contact. It makes sense to contact more customers than fewer even if some of them are not likely to open an account. The other models with high precision scores and lower recall will return fewer positive results, but most of their predictions will be correct when compared to dataset. This could be considered more useful by the bank, if they want to make more conservative predictions for how many actual term deposits will be made.

An ideal model would have high precision and high recall will return many results, with all results labeled correctly. I do believe that with more evaluation and fine-tuning of the training hyperparameters I could improve the other evaluation metrics of these models. I do not believe that the logistic regression model is the absolute best model that could be produced for this problem, especially since it still has a rather low accuracy and precision compared to other models built. However, I do believe that out of these 7 models this one is closest to being the most useful to the bank in determining which customers they should contact first in their next term deposit marketing campaign.

9 Feature Importance and Reduction

To identify a feature's relative importance on the model's accuracy I built 26 models, one for each feature in my dataset. These models each took only one input feature to see the effect this feature has on the validation accuracy of the model. I then graphed the results to depict the relationship between each input feature and the model's validation accuracy.

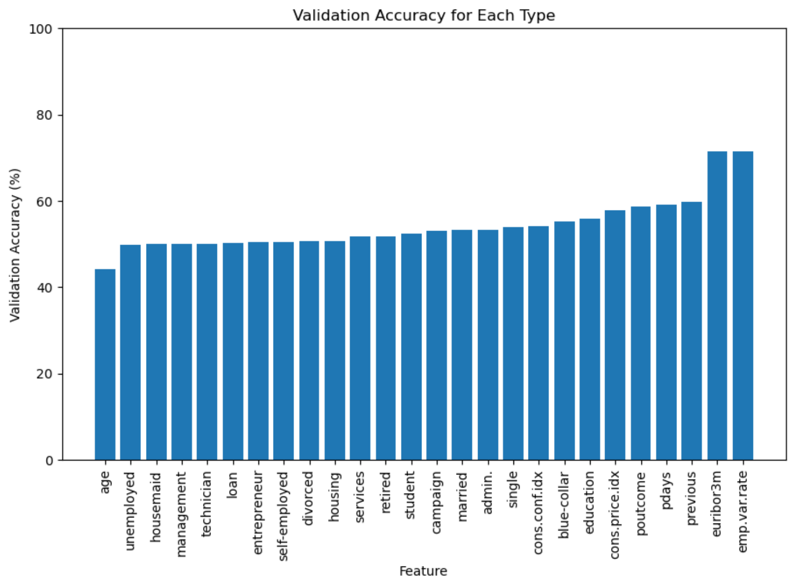


Figure 7: Validation accuracy of the models built with just one input feature to determine a feature's relative importance to the model.

Using the results of the relative importance experiment I then set up another series of 26 models. These models dropped features one by one, starting with the feature that had the lowest relative importance to the model. I was looking for features that might have added noise or bias to my model and wanted to see if the model I built could be improved by removing these features.

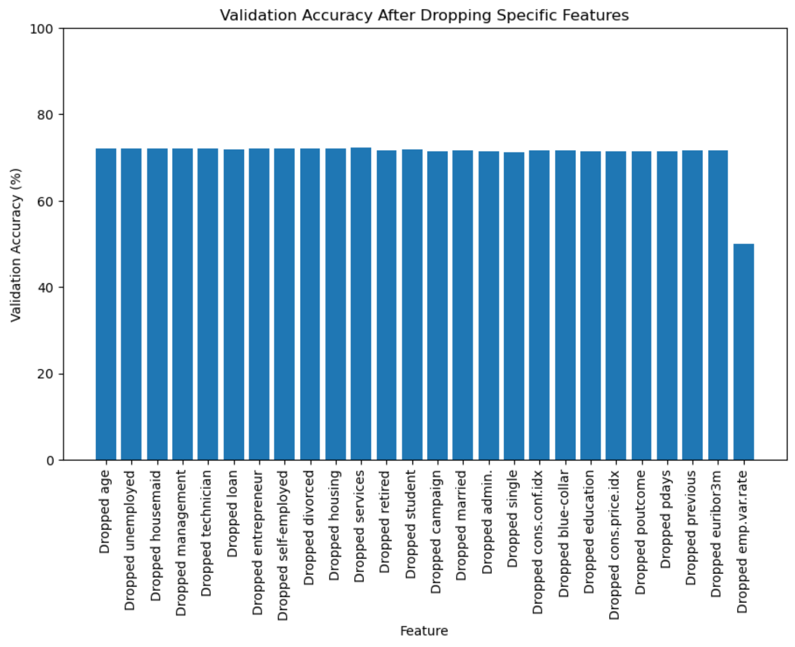


Figure 8: This graph illustrates the impact removing certain features has on the model's validation accuracy. The accuracy of the model is depicted after each iteration where features were removed one by one in order of least importance.

The results of these experiments were surprising. My initial hypothesis was the model was struggling with performance due to potentially noisy data or biased data. However, looking at the graphed results of dropping the low significance columns it does not appear that any particular column was adding severe noise or bias to the model. Overall, little changed in the accuracy of the model as features were removed from the models. The validation accuracy for the most part stayed about the same as the initial model built in Phase 3 which had a 71.6% validation accuracy. The only significant change in the model occurs when the emp.var.rate feature was dropped. This is because the model had no input features available to learn from so the model simply had to guess what the output would be. With basic probability it ended up with a 50% accuracy rate because this is a binary classification problem, there are only 2 potential output values it can guess.

I suspect the minimal feature impact is because as seen earlier in Phase 1 none of these features had a strong correlation with the output variable, so none of them have a strong impact on the model. I believe that if there were features with a stronger correlation to the output variable then this model would potentially perform better overall and we would have seen a greater change in this phase. At the same time this was a good exercise because I'd questioned in Phase 1 if I should have removed the poutcome and pdays features from the model to prevent noise. Considering the results here it appears that removing them would have not drastically changed my model results as initially thought.

10 Future Improvements

After completing this project I wonder if reducing the features further would have changed the model's results. For example, reducing the total job categories from 11 specific individual categories such as 'housemaid' and 'student' to fewer more broad-ranged categories such as grouping the 'unemployed', 'student', and 'retired' instances together as one and grouping the 'services' and 'technician' categories as another. I think that fewer job categories based on grouping similar job type/income ranges might help the model make better predictions. I also think that the model would have benefitted from having age simplified into a few age brackets such as ages 18-25, ages 26-40, and ages 41-60 instead of keeping it as a continuous numeric variable. Finally, some other data that the bank marketing team should consider trying to acquire is adding either monthly or annual income grouped into range brackets.

11 Conclusion

Ultimately it is extremely difficult to predict if a bank client will open a term deposit account or not. The dataset was large and had a comprehensive list of features but ultimately the features measured do not appear to have a strong correlation to whether a client decides to open a term deposit account or not. Unfortunately, none of the models performed particularly well. I believe this is because most of the features did not have strong feature relevance or correlation to the output variable. These models were initial drafts though and with further study, data, and fine-tuning could become very useful depending on the goals of the user.

I chose to study the Logistic Regression model further because I believed that this problem was best solved by a model that returned more potential positive results than negatives because I interpreted this data to be used by the marketing team to determine which bank clients they should contact first. In this scenario, it is logical to want to produce a larger pool of potential contacts for the marketers. However, if the bank wanted to use this data for more precise and conservative predictions of how many loans they should approve then they might want to review and analyze one of the other models such as the 16-8-1 neural network model that has a higher precision score for fewer positive predictions.

References

[AWS, 2023a] AWS (2023a). Model fit: Underfitting vs. overfitting - amazon machine learning.

[AWS, 2023b] AWS (2023b). What is overfitting? - overfitting in machine learning explained - aws.

[Brownlee, 2020] Brownlee, J. (2020). How to implement baseline machine learning algorithms from scratch with python.

[codebasics, 2018] codebasics (2018). *Machine Learning Tutorial Python - 6: Dummy Variables One Hot Encoding*. YouTube.

[Dolphin, 2022] Dolphin, R. (2022). Overfitting in ml: Avoiding the pitfalls.

[Herbas, 2020] Herbas, J. (2020). Deep learning neural network: Complex vs. simple model.

[Keldenich, 2021] Keldenich, T. (2021). Recall, precision, f1 score - simple metric explanation machine learning.

[Kumar, 2021] Kumar, S. (2021). 7 ways to handle missing values in machine learning.

[Online, 2023] Online, S. (2023). Handling categorical variables with one-hot encoding - shiksha online.

[Singh, 2023] Singh, S. (2023). Title: Label encoding and one-hot encoding for data preprocessing.

[Soni, 2023] Soni, B. (2023). Topic 1: Standardization and normalization.

[Vidhya, 2023] Vidhya, A. (2023). Feature selection techniques in machine learning (updated 2023).

[Whiteboard, 2023] Whiteboard, L. W. (2023). 8 tips on how to choose neural network architecture.

[Wohlwend, 2023] Wohlwend, B. (2023). Converting categorical data into numerical form: A practical guide for data science.