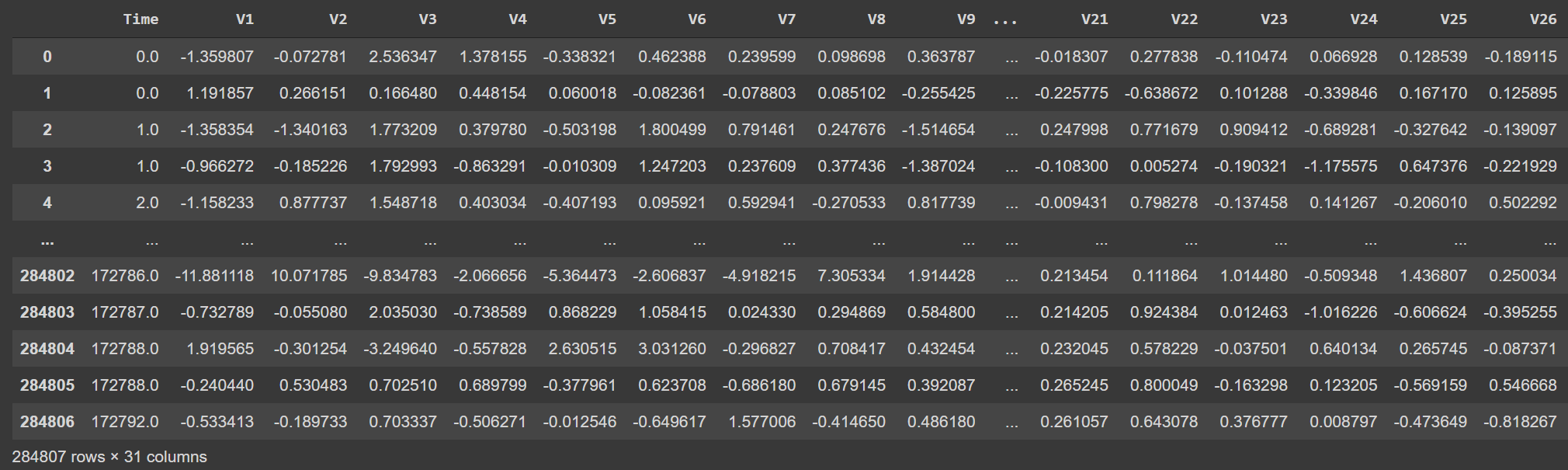
**Machine Learning Final Report**

Vishal Patel

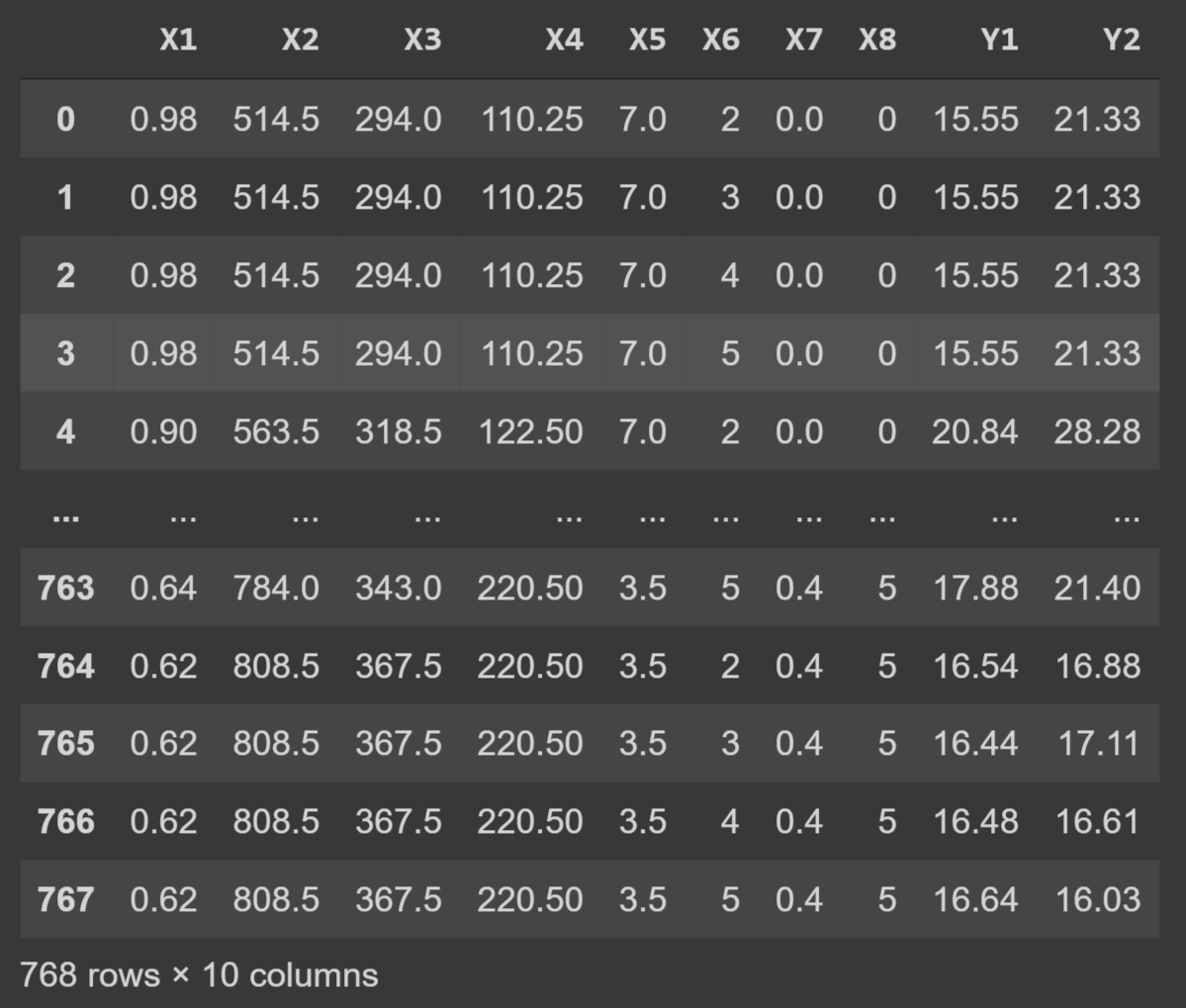
**Introduction**

This is a machine learning final project given to me by my professor and it will cover two different datasets which I will mention later on. The overall objective of the project is to read and analyze two different datasets using three distinct different machine learning algorithms. One dataset is a classification dataset and the other is a regressing dataset. Therefore, each will have their respective algorithms we learned in class. I will be going over the machine learning algorithms I chose and analyzing the results based off the models I coded to achieve high results. This is a first step for me in the right direction and it will highlight my extensive knowledge of machine learning through the use of six different algorithms. Below I will list the algorithms I chose for each of the datasets.

1. Credit card Fraud Detection Dataset (Classification)
   1. Logistic regression
   2. Random Forest Classifier
   3. Decision Tree Model



1. Energy Dataset (Regression)
   1. Neural Network Regressor
   2. Linear Regression
   3. Polynomial Regression



**Classification Dataset: Credit Card Fraud Detection**

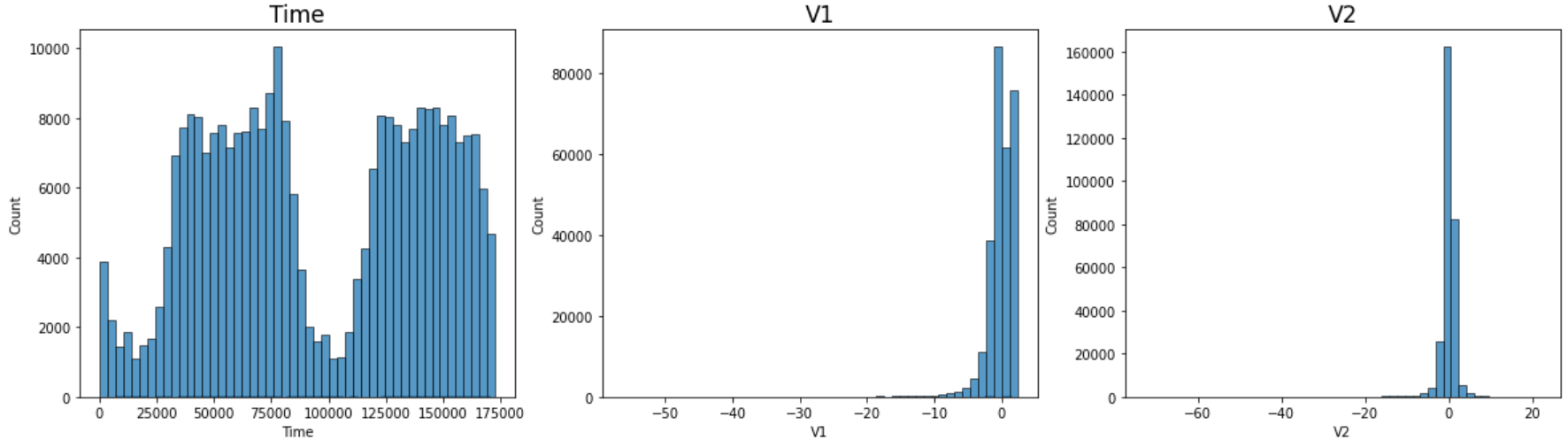
This dataset is a classification dataset with numerical values across twenty-eight different columns of data. It displays the number of samples that are fraudulent versus non fraudulent and the dataset itself contains a grand total of 285,000 samples of data. below I am going to list the methodology that I took to breakdown this data set and analyze the data.

1. Load data and import packages
2. Explore the dataset
3. Preprocess the dataset
4. Split the dataset into training and testing
5. Build the three classification models

The first thing I did was load the data and imported the necessary packages needed to run the program. I did add more packages as I went along with the code but I imported most of them at the top even though in machine learning, you import packages as you write the code in different cells. Printing the dataset showed me that I had twenty-eight columns to work with along with a column called amount. There were two more columns such as time and class but I knew off the bat that they did not have significance in this dataset and with the models that I was going to build.

They were a few things that I wanted to explore before I jumped into making the models and so I wanted to figure out how many samples were fraudulent and non-fraudulent. I wrote three lines of code displaying this information. The class column was binary classification for fraud versus non-fraud and therefore, I set each fraud samples equal to 1 and non-fraud equal to 0 so I can find out how many samples are in each category. The results from this were that the number of fraudulent transactions was 492 and non-fraudulent was 284,315.

Then I checked for any null values and there was not any so this was a good indication that everything would go smoothly. Since there were twenty-eight different columns of data, I wanted to print a graph for each of the columns which is what I did. Below is an example of the graphs.



I moved onto preprocessing the data and this is where I took care of the major imbalance in the data with the number of samples that were fraudulent versus non-fraudulent. I printed out the imbalance and then I used a method called resampling. This method will allow me to set the number of samples equal to non-fraud samples so it balances the dataset for accurate results in my models. The two images below are the before and after resampling happened. The imbalance is clearly visible and once this was done, I set my x and y variables while also printed out their values. The key significance of me resampling the data is in the beginning, I labeled my variable for reading my data “metadata” and now after resampling the data, I will have to use the variable called “metadata\_upsampled” because it contains the balanced data.

Shape, square

Description automatically generatedLogo

Description automatically generated with low confidence

I split the data using train, test, split and then I used standard scaling to normalize the data and this will help run the algorithms faster. I jumped right into the models I made and got promising results from all my models. Below is a chart showing my results for f1 score, precision score, recall score and accuracy score for all my models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy Score | Precision Score | Recall Score | F1 Score |
| Logistic regression | 95% | 97% | 92% | 94% |
| Random Forest | 99% | 99% | 100% | 99% |
| Decision Tree | 99% | 99% | 100% | 99% |

From the table, its clearly visible that I have put in extensive work into preprocessing and training the data because the only way for me to achieve such accuracies is when the training data that is supplied to the testing, it fits perfectly so the model performs perfectly with the numerical data. After doing research, achieving such accuracies is not unheard of and I was doubting myself when I saw these results but once I did the research, it was evident that I have overfitted my data which is why I got these accuracies. Overall, I did achieve what I wanted to with the dataset and that was to make sure I can predict whether or not there is a fraudulent transaction happening or not. The models speak for themselves and I was able to have accuracies that support my work.

**Regression Dataset: Energy Efficiency**

The regression dataset has to do with cooling and heating loads. There is picture of a building along with a excel file which contains columns from x1 to x8 and two target columns y1 and y2. The goal of this dataset is to predict the heating and cooling load use regression models. The dataset give does not label the x values but there is another similar dataset that labels them which I did not use because I do not think it is necessary to do extra work. The process that I took for this data is labeled below.

1. Import packages and reading dataset
2. Exploring the dataset
3. Preprocessing
4. Splitting y1 and y2 into two data frames
5. Models for y1
6. Models for y2

Chart, histogram

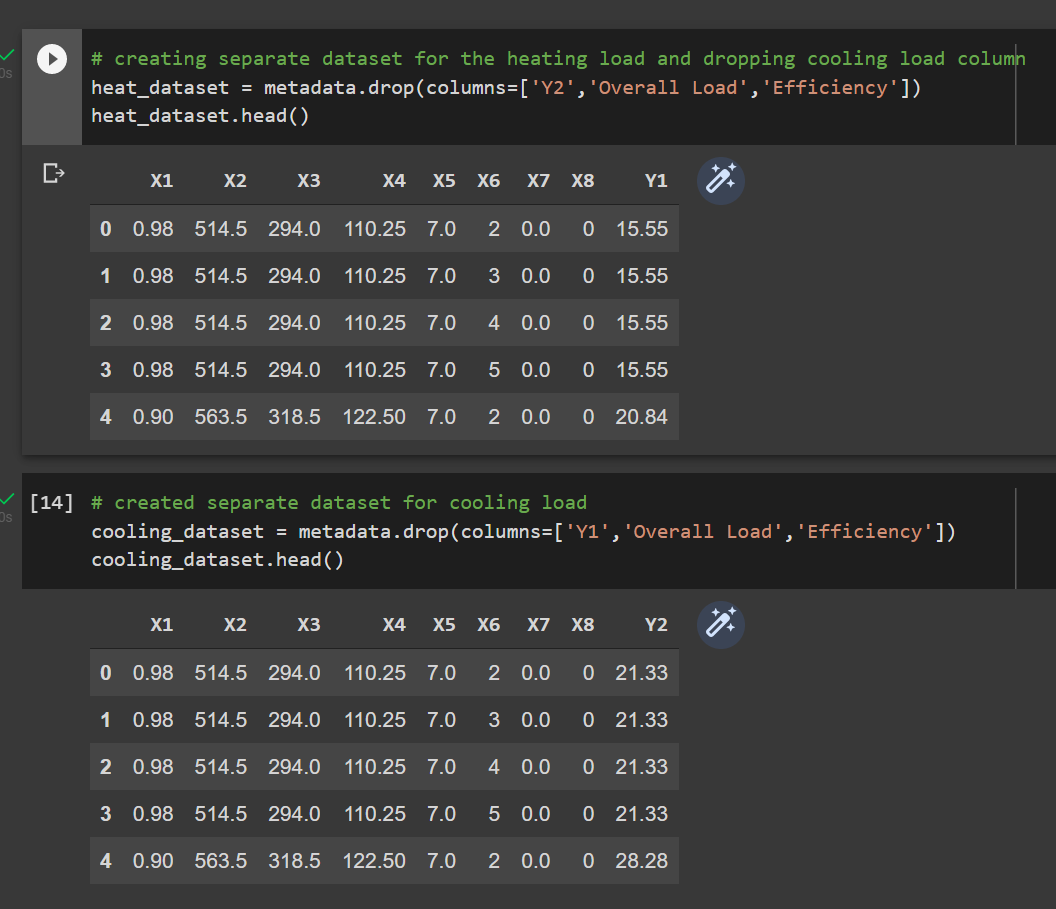
Description automatically generatedGraphical user interface, treemap chart

Description automatically generatedAfter loading the data, it was evident that there were eight features with two targets. The dataset had a total of 768 samples and the very next thing I decided to do was explore the dataset with graphs and tables. Below is a graph with the heating and cooling loads combined into a histogram which shows us any similarities or differences. Then I went on to find any null values and thankfully, there was not any null values.

The heatmap that I have generated above shows us distinctive features and values as well as highs and lows for the dataset. Since this dataset has to do with heating and cooling loads, a confusion matrix would not be of any use to me but a heat map can show me distribution of the data. Since all values are one’s going diagonal, it shows us that this is a good indication and nothing wrong is happening within the dataset.

Chart, bar chart

Description automatically generatedI amended the data and I wanted to do some more exploring as well to get better accuracy. I added an overall load column which combines y1 plus y1 which then allows me to add a column called efficiency. The reason behind this is because I wanted to see how many samples were high, low, and average. After adding the necessary columns, I did metadata[‘Efficiency’].values\_counts() to get 368 average samples, 207 high samples, and 193 low samples.

I chose a different approach to this dataset than normal students would and I felt this would be easier if I completed the first half of the problem and then the second half. I split the cooling and heating loads into two different datasets so I can predict the results separately for Y1 and Y2. For each data frame, I dropped Y1 or Y2 and overload and efficiency. The image here will explain how it was done and then how the dataset would look after I remove the columns.

The results are as follows in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Targets | Neural Network Regressor | Linear Regression | Polynomial Regression |
| Heating Load: Y1 | Mean Square Error: 3.15  Variance Score: 90% | Mean Square Error: 2.97  Variance Score: 91% | Mean Square Error: 0.80  Variance Score: 99% |
| Cooling Load: Y2 | Mean Square Error: 3.22  Variance Score: 88% | Mean Square Error: 3.2  Variance Score: 88% | Mean Square Error: 1.7  Variance Score: 96% |

I did three different regression models and the models produced excellent results as I was expecting them too. The neural network regressor was done over seventy-five epochs with a batch size of ten so this way it had enough epochs to run the data. I tried fifty epochs but I did not get what I was looking for so I increased it to seventy-five and it performed exceptionally good for the heating and cooling load. Linear regression is a basic regression model and overall, I was not expecting anything less than 85% accuracy from the variance score. My best model that performed better than I was expecting it to was polynomial regression and I have a few reasons behind why this performed the best. I got close to 100% accuracy for both models and originally, I was printing out the accuracy score instead of mean squared error and variance score. Once I printed out the mean squared error for polynomial regression, it showed me why the model performed so well.

The closer the mean squared error is to zero, the better your model is and, in this case, I have a mean squared error of 0.8 ad 1.7 which means that the model for the heating load is a perfect model. There is not anything better that model can do and the mean squared error states that. Overall, I got the results I was expecting for all my models and they predict the heating and cooling loads on an exceptional level.