Final Project Report for CS 184A/284A, Fall 2019

Project Title: Classifying Heart Disease using Neural Networks
Project Number: 26

Student Name(s)
Kashan Saeed, 69974920, khsaeed@uci.edu

1. Introduction and Problem Statement

Nearly half of the American population has some form of heart disease. Electrocardiograms, blood tests, and patient information are used to predict them. However, machine learning algorithms along with patent data and medical tests may help in more accurately predicting these diseases. Using 13 attributes: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca and thal from the UCI heart disease dataset[1] we trained a neural network to do a multiclass classification to predict if a patient has heart disease and, if so, which of the four types they have. Previous attempts have attempted to classify if a patient had heart disease or not (binary classification) and none have used neural nets. Our model was able to achieve a 66% accurate rate using a neural network with 2 hidden layers of size 10 each despite the small and very unbalanced dataset.

2. Related Work:

Previous methods have attempted to do a binary classification predicting whether a person has heart disease. A literature survey was done on previous research works. The number of attributes that were used for the final model ranged from three to eleven, and methods like Naïve Bayes classifier (NB), logistic regression, support vector machine (SVM), sequential minimal optimization (SMO), k-nearest neighbor (KNN), and many others were used[2]. The UCI repository has four datasets within their heart disease dataset; different research used different combinations of those four or supplemented it with other datasets.

We evaluated our neural network's performance while varying the features and datasets of its input, its layers, epochs, batch sizes, and hidden units to find the most accurate model and important features. We also did a multi class classification predicting a certain heart disease or no heart disease instead of a binary classification predicting the existence of a disease.

3. Data Sets

The datasets I used for this model can be found in the UCI machine learning repository and is called the heart disease dataset[1]. The dataset has 75 features and contains data from four different places, Cleveland, Hungary, Switzerland, and Long Beach. It also has 5 classes, 0 being no disease and 1 – 4 being different types of diseases. So far people have only used 13 of its features, those being age, sex,

[1] https://archive.ics.uci.edu/ml/datasets/heart+disease
[2] https://www.ijrte.org/wp-content/uploads/papers/v8i2S3/B11630782S319.pdf

cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca and thal; our input data used these or even less.

There are many problems with this dataset. Even though, when combined, the four sets have a total of 920 samples, many of the rows have missing data and, in some datasets, whole features are gone. The dataset is also very imbalanced which will lead to lower accuracy in predictions.

Database	Class 0	Class 1	Class 2	Class 3	Class 4	Total
Cleveland	164	55	36	35	13	303
Hungarian	188	37	26	28	15	294
Switzerland	8	48	32	30	5	123
Long Beach VA	51	56	41	42	10	200
Total	411	196	135	135	43	920

Fig 1. Table showing amount of data per class in each dataset

The table above is showing the amount of data for each class in each database. The ratios between class 0 and class 4 in terms of the amount of data is almost 10:1, and even between class 0 and class 1, which is the biggest class, the ratio is 2:1. Due to this imbalance, the model will be biased to class 0. Later, we will discuss the techniques we used to address this issue.

	age	sex	cp	trestbps	chol	fbs
Cleveland						
Hungarian	0.3 %	0.3 %	0.3 %	0.7 %	8.1 %	3.1 %
Switzerland				1.6 %	100 %	61 %
Long Beach VA				28.5 %	28 %	3.5 %

	restecg	thalach	exang	oldpeak	slope	ca	thal
Cleveland						1.3 %	0.7 %
Hungarian	0.7 %	0.7 %	0.7 %	0.3 %	64.7 %	99 %	90.5 %
Switzerland	0.8 %	0.8 %	0.8 %	4.9 %	13.8 %	95.9 %	42.3 %
Long Beach VA	0.5 %	26.5 %	26.5 %	28.0 %	51.0 %	99 %	83 %

Fig 2. Tables showing the percentage of data that is missing per feature in each dataset

The tables above show how much percentage of the 13 features each dataset is missing. Empty cells indicate that 0% of the data for that feature is missing. Even though the datasets combined have 920 rows, many of them will be thrown out depending on which features are chosen for the model’s input.

We experimented and found a good tradeoff between which features to keep and the number of samples in the dataset.

4. Description of Technical Approach

I broke up my approach into two parts, the first being understanding, preprocessing, and manipulating the data and the second related to applying the machine learning model to the inputted datasets.

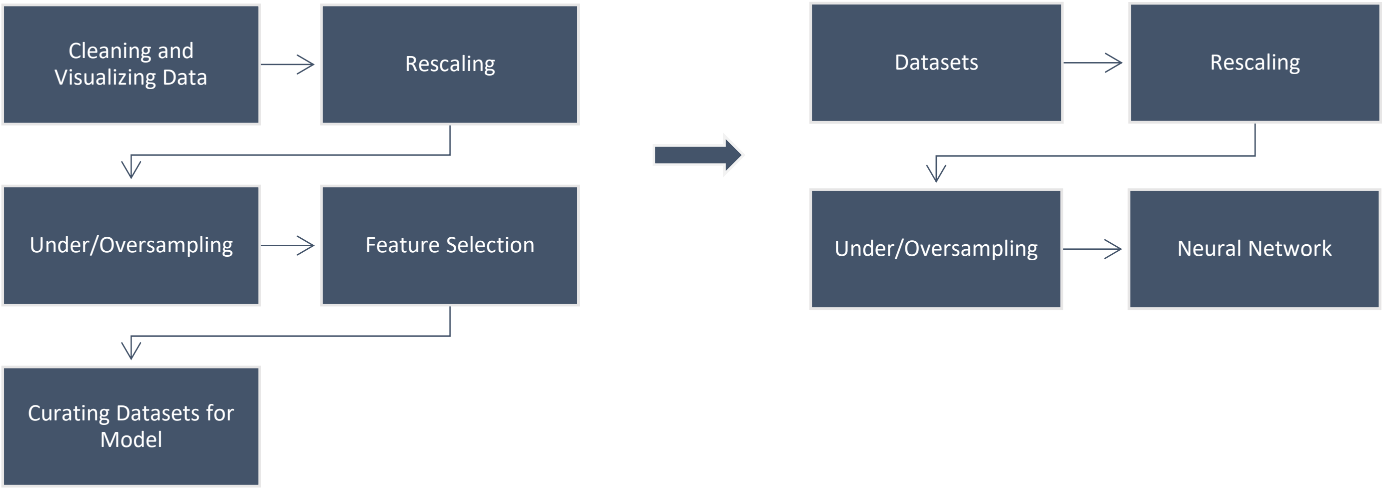


Fig 3. Diagram depicting technical approach

I began with getting a visual overview of the datasets by using pandas profiling[3], which is a python module that creates a visual report of the dataset containing information like the distribution of each column, the types of features, the number of missing values, correlation between features, and much more. Using this I was able to get a good idea of how unbalanced the dataset is, and how many values are missing. The two tables in the dataset section were formed from information from these reports. Because the dataset had a lot of missing values, we had to curate a dataset to give to the machine learning model that had a good balance between keeping certain features and having a lot of samples. This is because throwing out whole features may remove a lot NaN's from the data and allow us to keep those samples. Keeping a feature may mean that many samples who have missing values for that feature will get thrown out. To curate a balanced dataset, we will have to find which features are valuable enough to keep despite the cost that may come with keeping them. Additionally, because the dataset is very unbalanced, it will have to be rebalanced before applying any feature selection techniques.

The features of the dataset contain a wide variety of different ranges of values and are also of many types. I used a MinMaxScalar[4] from the scikit-learn library to rescale the data to a value between 1 and 0 and it also had the advantage of preserving the shape of the dataset. I did not use StandardScalar[5] as changing the data distribution to a normal distribution was not necessary for a neural network. Doing feature selection will reduce the dimensionality of the dataset and will allow us to have more samples by remove columns that have a lot of missing values, but to do that the dataset would need to be balanced. In order to rebalance the dataset, I employed one oversampling technique and seven undersampling techniques, and also did a combination of over and undersampling. For oversampling I tested Synthetic Minority Oversampling Technique (SMOTE), and for undersampling I tested NeighbourhoodCleaningRule, TomekLinks, ClusterCentroids, NearMiss, EditedNearestNeighbours, CondensedNearestNeighbour, OneSidedSelection, and RandomUnderSampler all from the imblearn

[3] https://github.com/pandas-profiling/pandas-profiling
[4] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
[5] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

API[6]. Five of the undersampling techniques led to a well-balanced dataset and their outputs were used as inputs for feature selection.

Three feature selection techniques were used: univariate selection, feature importance, and a correlation matrix with a heatmap. For univariate selection we used a chi-squared statistical test and implemented it using the SelectKBest[7] class from the scikit-learn library. Feature importance[8] is an inbuilt class within tree-based classifiers and returns a score for each feature of the data, a higher score indicating relevancy to the output variable. The correlation matrix was later thrown out as it only showed the relationship for numerical values even though many features were categorical.

Ultimately three datasets were curated that would function as the input for the machine learning model. Each having different number of features so that we could test the tradeoff between having more features verses having more samples.

The machine learning model used is a multilayer neural network implemented through the PyTorch library[9] and using the ReLU activation function. The three datasets functioning as the input for this model will first be rescaled. Each dataset will have five different undersamplers applied to it. Each undersampled dataset will then be trained and tested on a neural network. Each neural network will have 1, 2, and 3 hidden layers. For each hidden layer there will be 4, 8, 10, 12, 14, and 18 hidden units, and for each hidden unit the number of epochs will be 5, 10, 20, 40, 70, 100, 200, and 400. The model will be trained and tested on every possible combination of these parameters and datasets, and the best performing dataset, parameters will be chosen for the best model.

4. Software

The preprocessing module:

1. Cleaning and Visualizing the data
    a) Code that I wrote
        - The dataset itself was not processed very well. Some parts of it used “?”’s to indicate missing values and some 0’s. Much of the cleaning was done manually by looking at the values of the dataset and using the replace function from the pandas library.

    b) Code from other people
        - Pandas profiling module was used to visualize the datasets
2. Rescaling
    a) Code that I wrote
        - All four datasets were combined and any samples with missing values were removed. The dataset was then rescaled.

[6] https://imbalanced-learn.readthedocs.io/en/stable/api.html

[7] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest

[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[9] https://pytorch.org/docs/stable/nn.html

b) Code from other people
    - MinMaxScalar from scikit-learn was used to rescale the data

3. Under/Oversampling
    a) Code that I wrote
        - Using the rescaled dataset, the number of samples for each class was balanced using various sampling methods. Datasets that came from well balancing sampling techniques were passed on to the next module, there were five in total.

    b) Code from other people
        - All the under/oversampling methods came from the imblearn API

4. Feature Selection
    a) Code that I wrote
        - Feature selection algorithms were ran on each of the balanced datasets, these were used to rank the importance of each feature.

    b) Code from other people
        - The code for all three feature selection techniques came from Raheel Shaikh[10]

5. Curating Datasets
    a) Code that I wrote
        - The importance of each feature was considered when deciding whether to throw it out for the benefit of keeping more samples or to keep it and lose some samples.
        - Ultimately three datasets were curated by combining the original four datasets and dropping certain features and rows from each. The different datasets will allow testing the tradeoff between the features selected vs the samples lost.

The model selection module:

1. Loading and assembling the curated datasets
    a) Code that I wrote
        - Loaded all the datasets and combined them with the proper features as decided in the previous module.

2. Undersampling Function
    a) Code that I wrote
        - I created a function that would return a list of tuples, each tuple containing a balanced version of the dataset. I used five undersamplers, so it returned a list of five tuples.

    b) Code from other people
        - All the under/oversampling methods came from the imblearn API

3. Dataloader Function
    a) Code that I wrote
        - Created a function that takes in the testing data, validation data, and batch size and returns them as a tuple of DataLoaders.

    b) Code from other people
        - Torch library

[10] https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:~:text=Feature%20Selection%20is%20the%20process,learn%20based%20on%20irrelevant%20features.

- Much of this section was inspired from Professor Xiaohui Xie's Jupyter notebook on convolutional neural networks[11].

4. Model Class
    a) Code that I wrote
        - Three neural network constructors were created, one with one hidden layer, one with two, and one with three.
    b) Code from other people
        - Much of this section was inspired from Professor Xiaohui Xie's Jupyter notebook on convolutional neural networks[11].
5. Dataset Selection
    a) Code that I wrote
        - This section allowed me to pick which of the three datasets I wanted to run and split and rescaled them.
    b) Code from other people
        - The scikit-learn library for scaling and splitting
6. Hyperparameters Section
    a) Code that I wrote
        - This section allowed me to pre-pick which values I wanted to test for each hyper-parameter.
7. Training and Testing
    a) Code that I wrote
        - This module is where everything was combined and consisted of four for loops just to test different variations. The first loop was for each variation that came from balancing the datasets, the second for the numbers of hidden layers I wanted to test, the third for the amount of hidden units, and the fourth for the different epochs I wanted to test.
    b) Code from other people
        - The code for training and testing the model was taken from Professor Xiaohui Xie's Jupyter notebook on convolutional neural networks[11].

5. Experiments and Evaluation

This section can be divided into a preprocessing and model selection section:

When preprocessing the data the main experimentation was with different undersampling algorithms and with feature selection. After combining all the datasets, rescaling the, and removing any invalid value we were left with a dataset containing 300 samples. This will be referred to as the original dataset (OD) for now. Then eight different undersampling techniques; RandomUnderSampler (RUS), CondensedNearestNeighbour (CNN), NearMiss (NM), TomekLinks (TL), EditedNearestNeighbours (ENN), NeighbourhoodCleaningRule (NCR), OneSidedSelection (OSS), and ClusterCentroids (CC) were applied to it and here are the results.

Class	0	1	2	3	4
OD	160	56	35	35	13

[11] https://github.com/xhxuciedu/CS284A/blob/master/convolutional_neural_net.ipynb

RUS	13	13	13	13	13
CNN	18	16	15	16	13
NM	13	13	13	13	13
TL	143	36	24	23	13
ENN	92	56	35	35	13
NCR	141	20	2	4	13
OSS	40	27	23	20	13
CC	13	13	13	13	13

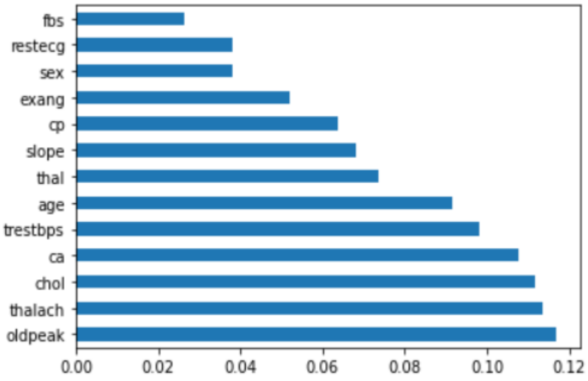
Fig 4. This table is showing how many samples of each are there in the dataset after it was balanced

RUS, CNN, NM, OSS, and CC returned well balanced dataset and so the balanced datasets that came from them were used in feature selection techniques.

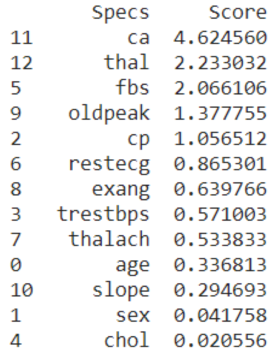
Though three feature selection techniques were used; univariate selection, feature importance, and a correlation matrix with a heatmap, the correlation matrix was thrown out as it only showed the relationship for numerical values even though many of the features were categorical.

For the other two feature selection techniques the results are in the tables below. CC could not be used for univariate selection as the dataset it returned contained negative values. Also because the dataset resulting from OSS was not as balanced a the rest the results of its feature selection algorithms were given less weight.

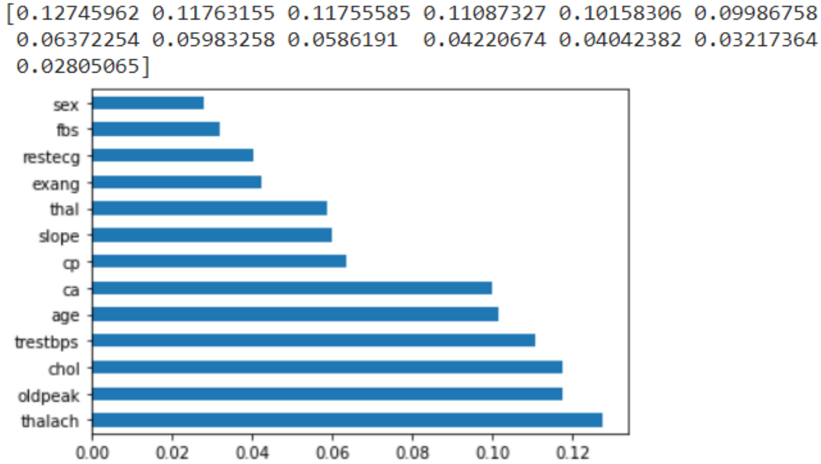
Balancing Algorithm	Univariate Selection	Feature Importance

RUS	`Specs Score`
	`11 ca 6.589372`
	`8 exang 5.214286`
	`12 thal 5.036723`
	`10 slope 3.700000`
	`5 fbs 3.090909`
	`9 oldpeak 2.651716`
	`6 restecg 2.000000`
	`2 cp 1.477707`
	`1 sex 1.200000`
	`7 thalach 0.535566`
	`3 trestbps 0.203142`
	`0 age 0.080486`
	`4 chol 0.070141`
		`[0.11679796 0.11364816 0.11175712 0.10766147 0.0980323 0.09174568 0.07375089 0.06809592 0.06364887 0.05209479 0.03821755 0.03806762 0.02648168]`
		

CNN

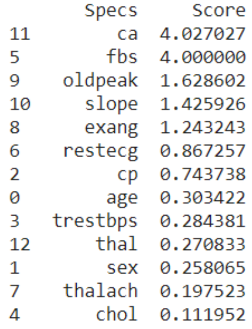


```
       Specs     Score
11        ca  4.624560
12      thal  2.233032
5        fbs  2.066106
9    oldpeak  1.377755
2         cp  1.056512
6    restecg  0.865301
8      exang  0.639766
3   trestbps  0.571003
7    thalach  0.533833
0        age  0.336813
10     slope  0.294693
1        sex  0.041758
4       chol  0.020556
```

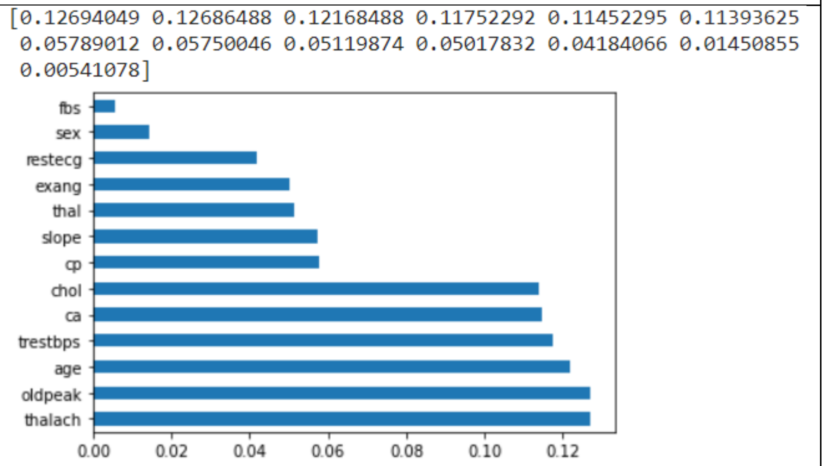


```
[0.12745962 0.11763155 0.11755585 0.11087327 0.10158306 0.09986758
 0.06372254 0.05983258 0.0586191  0.04220674 0.04042382 0.03217364
 0.02805065]
```

NM

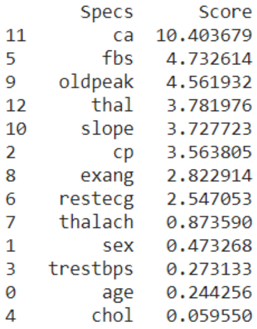


```
       Specs     Score
11        ca  4.027027
5        fbs  4.000000
9    oldpeak  1.628602
10     slope  1.425926
8      exang  1.243243
6    restecg  0.867257
2         cp  0.743738
0        age  0.303422
3   trestbps  0.284381
12      thal  0.270833
1        sex  0.258065
7    thalach  0.197523
4       chol  0.111952
```

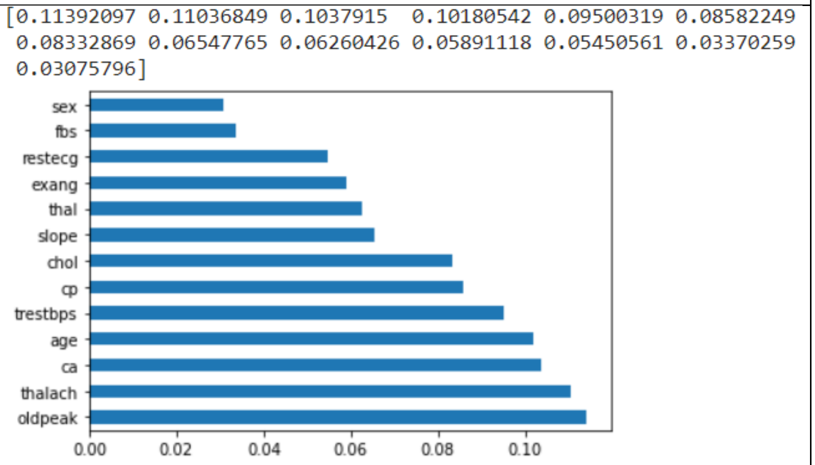


```
[0.12694049 0.12686488 0.12168488 0.11752292 0.11452295 0.11393625
 0.05789012 0.05750046 0.05119874 0.05017832 0.04184066 0.01450855
 0.00541078]
```

OSS



```
       Specs      Score
11        ca  10.403679
5        fbs   4.732614
9    oldpeak   4.561932
12      thal   3.781976
10     slope   3.727723
2         cp   3.563805
8      exang   2.822914
6    restecg   2.547053
7    thalach   0.873590
1        sex   0.473268
3   trestbps   0.273133
0        age   0.244256
4       chol   0.059550
```



```
[0.11392097 0.11036849 0.1037915  0.10180542 0.09500319 0.08582249
 0.08332869 0.06547765 0.06260426 0.05891118 0.05450561 0.03370259
 0.03075796]
```

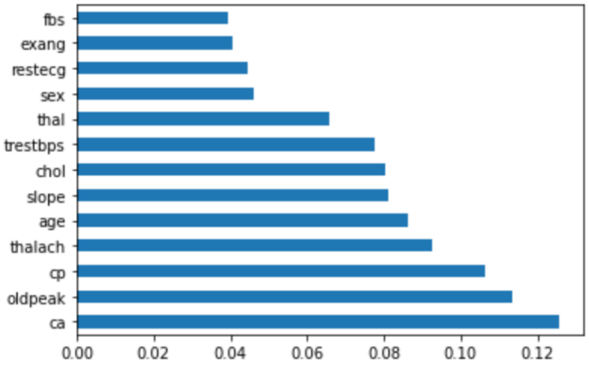
CC	N/A	[0.12554518 0.11328599 0.10622462 0.09272528 0.08630335 0.08100537 0.08045645 0.07763125 0.06583951 0.04610403 0.04469006 0.04063333 0.03955557] 

Fig 5.

When looking at the results from the feature importance technique consistently among the top six features are; thalach, oldpeak, chol, age, trestbps, and ca; and they are even according to some of the graphs quit a bit more important than the rest. In the middle are; slope, cp, thal, exang, and restecg; and the lowest which are significantly less important are fbs, and sex. It is interesting to note that in univariate selection there was more variation between all the balanced dataset but all had ca in the top, and fbs was usually ranked very high too. Due to their being more consistency the results given by the feature importance technique were given preference.

Looking back figure 2, the table showing how much of each feature was missing from the four datasets, notice that ca is missing in 3 of the datasets, chol missing fully in 1 dataset, slope missing partially in 3 of them, and thal also missing in 3 of them. ca was ranked very high in univariate selection and feature importance rankings, and chol in just feature importance rankings. thal was ranked medium or low so was dropped from all the datasets for a gain in samples. slope was also ranked medium so it would be interesting to see its effect if it were dropped form the dataset. chol was kept as it was among the top six most important features and only resulted in a loss of 1/10 of the samples, but because ca was deemed important yet resulted in the loss of ¾ of the samples, three datasets were created that would be tested in the model.

Dataset #1	Dataset #2	Dataset #3
- Drop ca, slope - Results in: - 558 Samples, 10 features	- Drop ca - Results in: - 392 Samples, 11 features	- Results in: - 297 Samples, 12 features

Fig 6

These three datasets seemed like a good way to test the features vs sample size balance and once plugged into the model we can see how effective each one is.

For selecting the hyper-parameters and layers for our neural network as mentioned in the software and technical approach sections we took the outputs from 5 undersampling techniques (RUS, CNN, NM, OSS, and CC) and applied them to a nueral network. For each neural network we tested 1, 2, and 3 hidden layers. For each hidden layer 4, 8, 10, 12, 14, and 18 hidden units, and for each hidden unit 5, 10, 20, 40, 70, 100, 200, and 400 epochs. We split the dataset into training and validation sets using a 25% split, making sure that the validation set had some data from each class to test with. Rescaling was applied to both and only the training data was rebalanced. The top ten results for each dataset are as follows.

Undersampler	NN Hidden Layers	Hidden Units	Epochs	Train Accuracy	Val Accuracy
OSS	1	12	10	51.24	60.53
OSS	1	12	70	55.37	60.53
OSS	3	8	200	60.33	60.53
CC	3	18	70	56	60.53
CNN	1	4	400	39.39	59.21
CNN	1	8	200	43.94	59.21
CNN	2	12	40	37.88	59.21
CNN	3	18	40	34.85	59.21
OSS	1	4	100	53.72	59.21
OSS	1	8	100	59.5	59.21

Fig 7. The top ten highest neural net model for dataset 3

Undersampler	NN Hidden Layers	Hidden Units	Epochs	Train Accuracy	Val Accuracy
CNN	1	4	200	40.28	51.72
CNN	2	8	100	35.42	51.72
CNN	2	14	20	34.72	51.72
OSS	1	12	40	55.43	51.72
CNN	1	4	70	34.72	50.86
CNN	1	4	100	36.11	50.86
CNN	2	10	200	43.06	50.86
OSS	3	8	100	58.91	50.86
CNN	1	8	100	39.58	50
CNN	2	8	400	34.03	50

Fig 8. The top ten highest neural net model for dataset 2

Undersampler	NN Hidden Layers	Hidden Units	Epochs	Train Accuracy	Val Accuracy
OSS	2	8	200	64.86	60.24
OSS	2	10	100	52.7	60.24
OSS	2	14	20	47.3	60.24
OSS	3	18	70	51.35	60.24
OSS	2	12	100	56.76	59.64
OSS	2	14	70	55.41	59.64
OSS	3	8	100	51.35	59.64
OSS	3	8	200	48.65	59.64
OSS	3	10	200	58.11	59.64
OSS	1	10	100	60.81	59.04

Fig 9. The top ten highest neural net model for dataset 1

From looking at these results the two most important things seem to be the dataset and the undersampling method. The highest dataset 2 was able to go was 51.72% while dataset 1 and 3 were able to go to around 60%. Another thing to note is that OSS and CNN are constantly among the highest performing models, this is vey surprising to me as I did not know that the undersampling technique could result in such consistent results. It also seems like the number of hidden layers, hidden units, epochs did not effect the accuracy much as there is a lot of variation in them among the top performers. I can not understand why dataset 2 performs the lowest. Dataset 1 dropped ca and slope, dataset 2 dropped ca, and dataset 3 did not drop those two, so it seems that using it should have resulted in an accuracy between dataset 1 and 3.

6. Discussion and Conclusion

The most important insight I gained from this project is that no matter how the machine learning model is structured if the dataset is not good it will not yield good results. This dataset had a lot of issues from the very beginning, it was imbalanced, had a lot of missing values, and had a low sample size. Even when the features were varied to increase the size of the dataset, due to the imbalance nature of the dataset undersampling would decrease the total sample size significantly, usually to around 20%. The low sample size would help explain why the undersampling technique played a significant part in the accuracy of the model, as a different technique would reflect the data in different ways.

If I oversaw a research lab I would focus the next year on collecting better and more data on the problem as that would lead to the most improvement in the accuracy of the model.

7. A separate page on *Individual Contributions*

Kashan Saeed

I did this project solo and thus worked on the whole thing.