CMP310 Fundamentals of Machine Learning Project Report [1]

Robbie Sangster 2003115

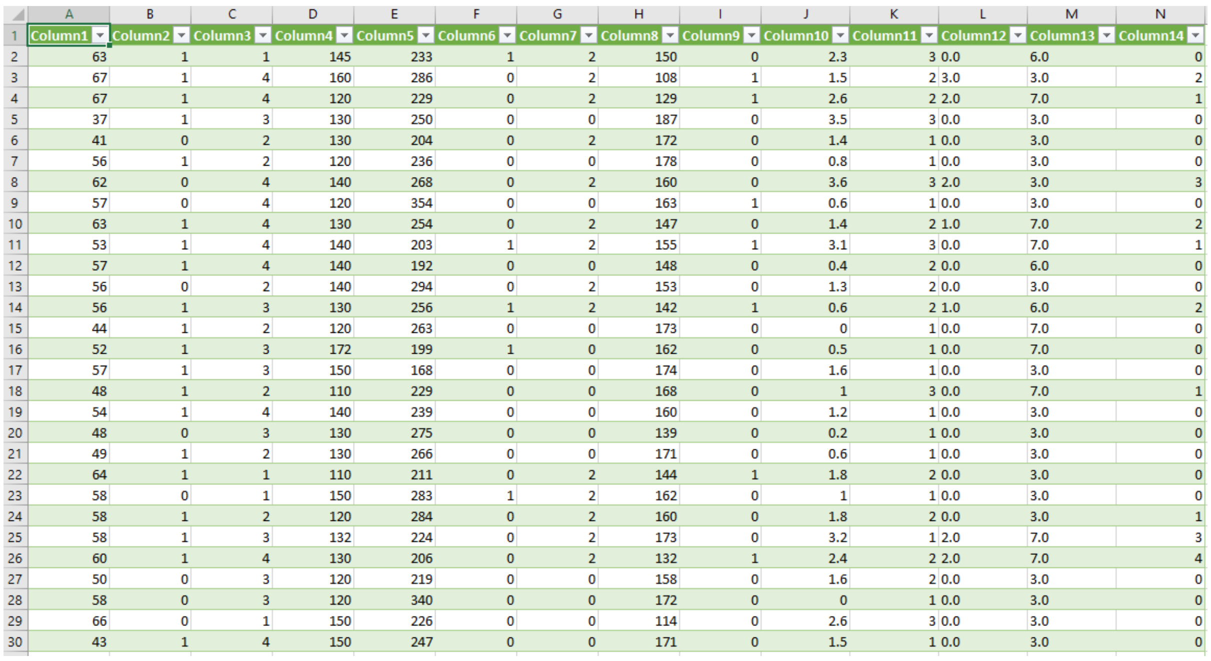
[1] Structure, formatting, spelling, grammar, and coherence (5%)

Introduction

Heart Disease is one of the leading causes of death in the world with a person dying every three minutes in the UK from it (British Heart Foundation, 2023). The goal of this project is to provide patients and doctors an accurate machine learning model to determine the likelihood of heart disease when provided with the patients data. The model will be a logistic regression binary classifier and will output either the absence or presence of heart disease within a patient. Logistic regression is commonly used by healthcare organisations to predict the likelihood of disease or illness (IBM, no date). The model will be trained and tested using data from the Cleveland heart disease dataset and will be used with three models. The first model will use all 14 data features provided in the dataset in order to receive an output. The second model will be created from analysing the correlation matrix in relation to the target variable 'condition' to determine which features contribute the most and least. The third and final model will be adapted from another researcher which involves grouping select features before fitting the model (Marangoz, B. ,2023). The model will be evaluated using its overall accuracy, precision, recall, f1 score and a confusion matrix. When evaluating the confusing matrix the goal is to have as many correct predictions as possible but when it comes to incorrect predictions the aim is to have more false positives than false negative. This is because we want people who use the model if given an output that is positive they would refer to a doctor for further information, however if they receive a false negative it is unlikely they will seek assistance. The final solution chosen is the third model, it provided the highest accuracy when compared the other models and an impressive ROC Curve which means minimal false positives.

Data Specification

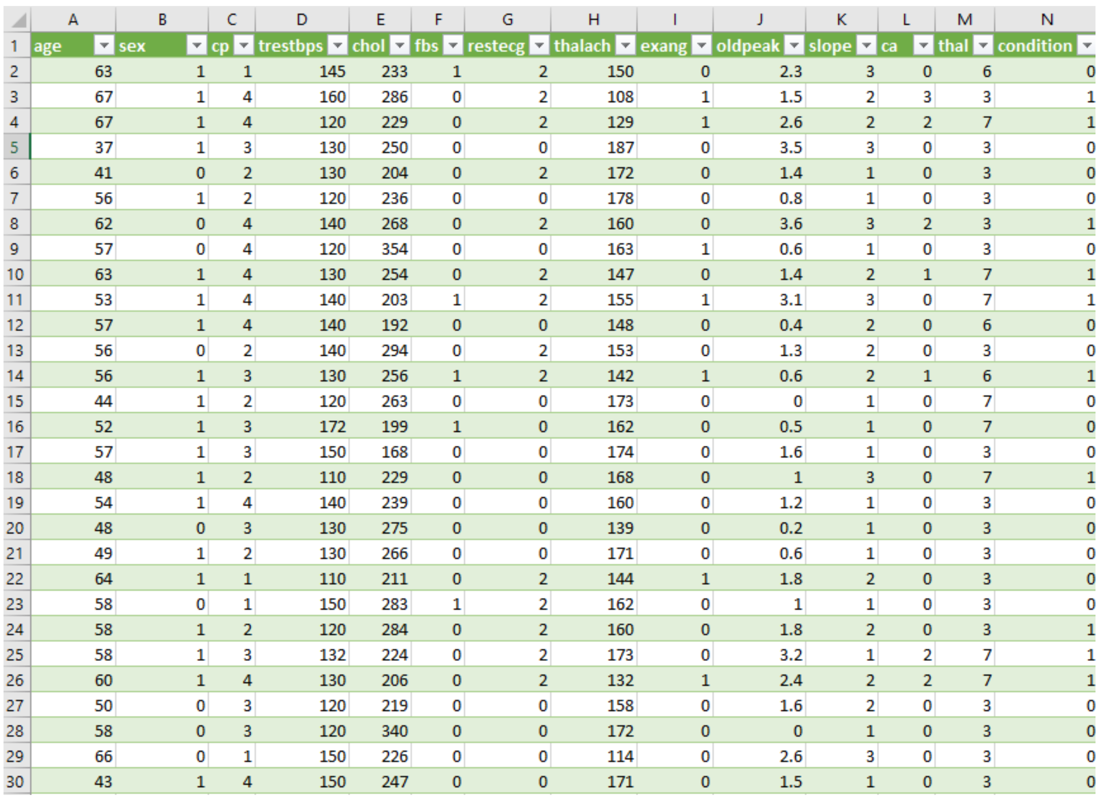
The dataset consists of 300 counts of patient data made up of 13 features and the target variable.(Detrano, R. ,1998). The data is made up of two files, firstly the data itself is from the processed.cleveland.data and its matching descriptor file called heart-disease.names file. The descriptor file explains all the features in depth and was used when adding labels to the raw data. The labels used all match the descriptor file with the exception of the final column titled 'num' which is the output data where 0 is absence and 1 to 4 is presence of heart disease. The column was named 'condition' in the model to make it clearer what it represents.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11	Column12	Column13	Column14
2	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
3	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2
4	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
5	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
6	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0
7	56	1	2	120	236	0	0	178	0	0.8	1	0.0	3.0	0
8	62	0	4	140	268	0	2	160	0	3.6	3	2.0	3.0	3
9	57	0	4	120	354	0	0	163	1	0.6	1	0.0	3.0	0
10	63	1	4	130	254	0	2	147	0	1.4	2	1.0	7.0	2
11	53	1	4	140	203	1	2	155	1	3.1	3	0.0	7.0	1
12	57	1	4	140	192	0	0	148	0	0.4	2	0.0	6.0	0
13	56	0	2	140	294	0	2	153	0	1.3	2	0.0	3.0	0
14	56	1	3	130	256	1	2	142	1	0.6	2	1.0	6.0	2
15	44	1	2	120	263	0	0	173	0	0	1	0.0	7.0	0
16	52	1	3	172	199	1	0	162	0	0.5	1	0.0	7.0	0
17	57	1	3	150	168	0	0	174	0	1.6	1	0.0	3.0	0
18	48	1	2	110	229	0	0	168	0	1	3	0.0	7.0	1
19	54	1	4	140	239	0	0	160	0	1.2	1	0.0	3.0	0
20	48	0	3	130	275	0	0	139	0	0.2	1	0.0	3.0	0
21	49	1	2	130	266	0	0	171	0	0.6	1	0.0	3.0	0
22	64	1	1	110	211	0	2	144	1	1.8	2	0.0	3.0	0
23	58	0	1	150	283	1	2	162	0	1	1	0.0	3.0	0
24	58	1	2	120	284	0	2	160	0	1.8	2	0.0	3.0	1
25	58	1	3	132	224	0	2	173	0	3.2	1	2.0	7.0	3
26	60	1	4	130	206	0	2	132	1	2.4	2	2.0	7.0	4
27	50	0	3	120	219	0	0	158	0	1.6	2	0.0	3.0	0
28	58	0	3	120	340	0	0	172	0	0	1	0.0	3.0	0
29	66	0	1	150	226	0	0	114	0	2.6	3	0.0	3.0	0
30	43	1	4	150	247	0	0	171	0	1.5	1	0.0	3.0	0

Fig 1: Raw data from processed.cleveland.data

The pre-processing steps for the raw data were as follows: Firstly the labels were added to each column. Logistic regression is a supervised algorithm and requires labelled data. The labels provided inside the descriptor file were: (In order) 'age', 'sex', 'cp', 'trestbps', 'chol,', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal' and the target variable 'condition'. As mentioned earlier the dataset contains missing values. This was represented in the file as '?'. These were present in the 'ca' and 'thal' column. The rows they were found it were removed in their entirety. This was done as later the correlation matrix identifies these as columns that relate to the target significantly and to help reduce inaccuracies when training the model. The other major change to the data was the target variable. The descriptor explains the column as 0 means no presence of heart disease and numbers 1 through 4 mean presence of heart disease. To make the development of the model simpler any value above 1 was changed to 1. As this is a binary classifier this reduces the steps required to get a meaningful result. The model will need to interpret values 1, 2, 3 and 4 as 1 and by doing it beforehand speeds up development for the software.



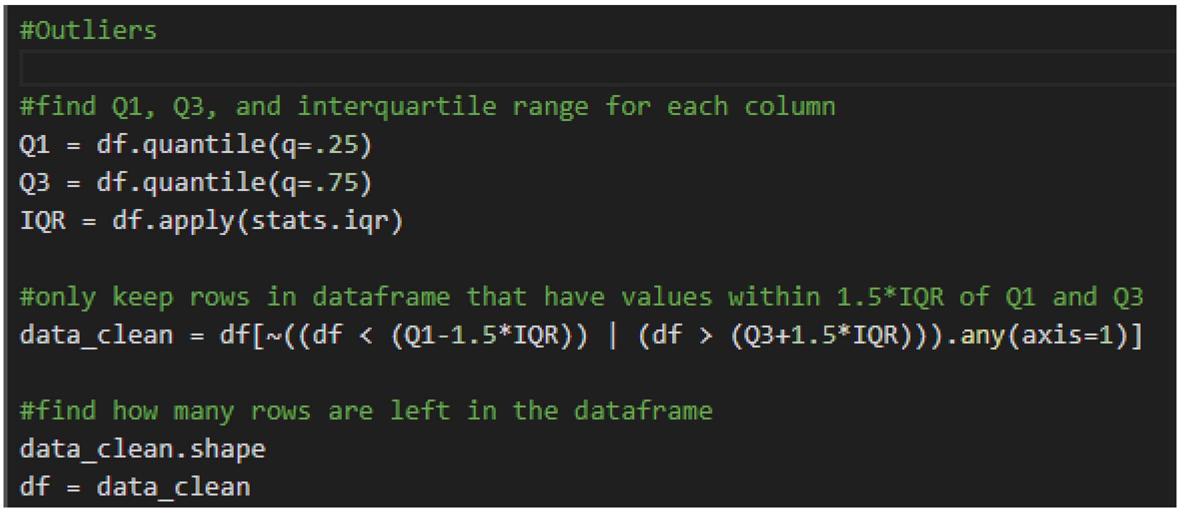
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
2	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
3	67	1	4	160	286	0	2	108	1	1.5	2	3	3	1
4	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
5	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
6	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
7	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
8	62	0	4	140	268	0	2	160	0	3.6	3	2	3	1
9	57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
10	63	1	4	130	254	0	2	147	0	1.4	2	1	7	1
11	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
12	57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
13	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
14	56	1	3	130	256	1	2	142	1	0.6	2	1	6	1
15	44	1	2	120	263	0	0	173	0	0	1	0	7	0
16	52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
17	57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
18	48	1	2	110	229	0	0	168	0	1	3	0	7	1
19	54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
20	48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
21	49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
22	64	1	1	110	211	0	2	144	1	1.8	2	0	3	0
23	58	0	1	150	283	1	2	162	0	1	1	0	3	0
24	58	1	2	120	284	0	2	160	0	1.8	2	0	3	1
25	58	1	3	132	224	0	2	173	0	3.2	1	2	7	1
26	60	1	4	130	206	0	2	132	1	2.4	2	2	7	1
27	50	0	3	120	219	0	0	158	0	1.6	2	0	3	0
28	58	0	3	120	340	0	0	172	0	0	1	0	3	0
29	66	0	1	150	226	0	0	114	0	2.6	3	0	3	0
30	43	1	4	150	247	0	0	171	0	1.5	1	0	3	0

Fig 2: Processed Data

Once the pre-processing is done three rows of data were selected at random to be used for test cases once the model is trained. These rows were stored in their own respective file and then removed from the overall dataset.

Methodology

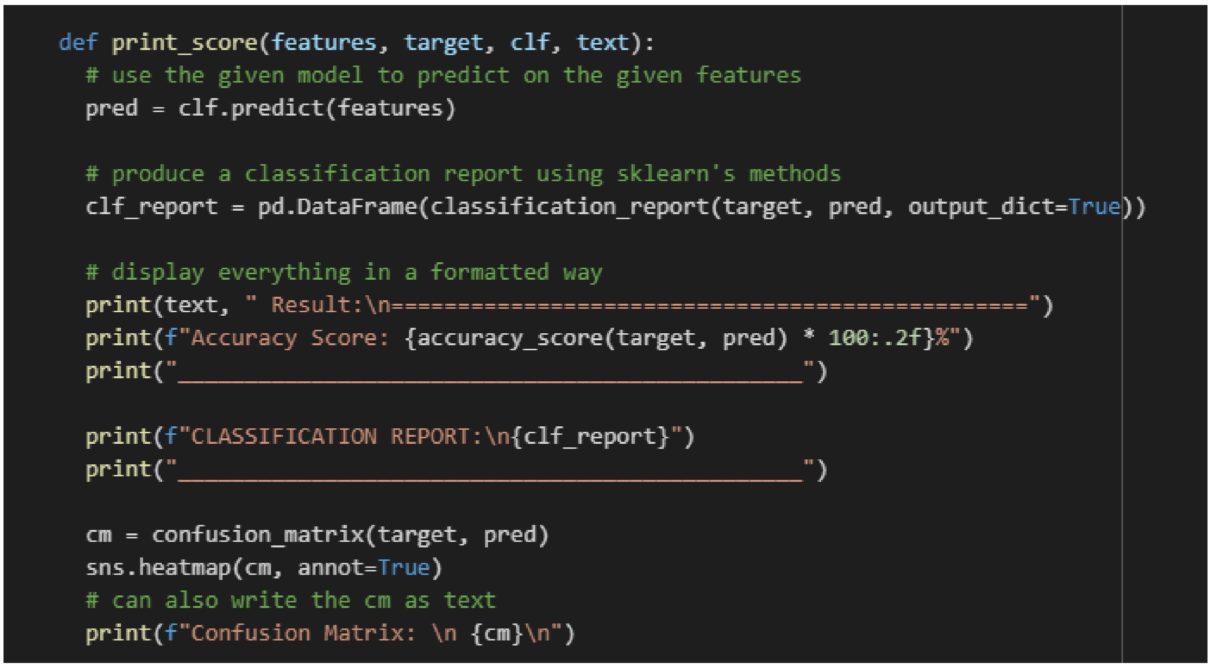
The first step of the project is to import the libraries and the Cleveland data. The main libraries used throughout each of the models are pandas, sklearn numpy and seaborn. Pandas read_csv is used to get the data into the code. To make sure our data pre-processing has been successful use the isnull() function to see if any missing values are still present. If none are identified then proceed to remove outliers. Create 2 variables called Q1 and Q3, in Q1 have the dataframe and use .quantile .25 and for Q3 use .75, see Fig 3. The interquartile range (IQR) should also be stored using the stats library function .iqr. Using all these variables remove the data out with 1.5 times the IQR and save it as the new dataframe. See Fig 3.



```
#Outliers

#find Q1, Q3, and interquartile range for each column
Q1 = df.quantile(q=.25)
Q3 = df.quantile(q=.75)
IQR = df.apply(stats.iqr)

#only keep rows in dataframe that have values within 1.5*IQR of Q1 and Q3
data_clean = df[~((df < (Q1-1.5*IQR)) | (df > (Q3+1.5*IQR))).any(axis=1)]

#find how many rows are left in the dataframe
data_clean.shape
df = data_clean
```

Fig 3: Removing Outliers

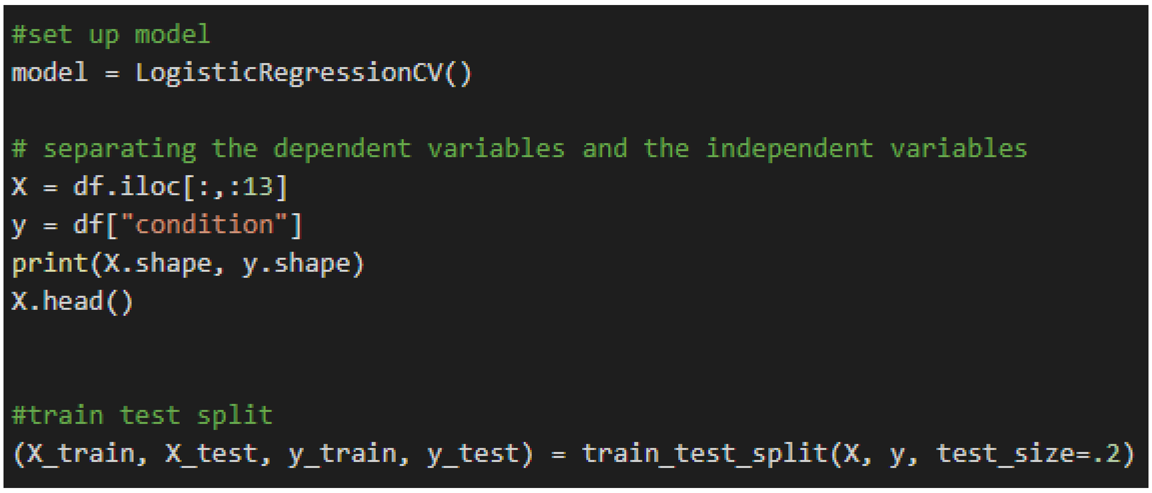
A helper function is also created at the start as it is used to calculate performance metrics such as accuracy, recall, f1 score and a confusion matrix. See Fig 4.



```
def print_score(features, target, clf, text):
    # use the given model to predict on the given features
    pred = clf.predict(features)

    # produce a classification report using sklearn's methods
    clf_report = pd.DataFrame(classification_report(target, pred, output_dict=True))

    # display everything in a formatted way
    print(text, " Result:\n================================================")
    print(f"Accuracy Score: {accuracy_score(target, pred) * 100:.2f}%")
    print("_______________________________________________")

    print(f"CLASSIFICATION REPORT:\n{clf_report}")
    print("_______________________________________________")

    cm = confusion_matrix(target, pred)
    sns.heatmap(cm, annot=True)
    # can also write the cm as text
    print(f"Confusion Matrix: \n {cm}\n")
```

Fig 4: Helper Function

The function takes in the X data, the y data the model and text to provide detailed metrics that can be analysed to improve model performance. A histogram is a good addition to machine learning models as it helps users understand how the data looks and the breakdown for each feature. Seaborn histplots for features with many values such as cholesterol or countplots for features with limited values like Chest Pain Type. With this completed the first model can begin. Each model operates in the same way with the only difference between each model is the data it handles. Declare the model from sklearn LinearRegressionCV(). CV stands for cross validation and further trains the model but creating subsets of the training data, of these subsets the one with the lowest generalisation error will then be used to retrain the entire train data set. This is used to flag problems such as overfitting or selection bias (Team, 2022). Then create the dependent and independent variables X and y, where X is the features and y is the target variable 'condition'. Train_test_split is used to create datasets for training and testing the model. The split is 80% training and 20% training. This is the standard ratio for datasets. The function creates 4 variables called X_train, X_test, y_train and y_test. The model is now ready to be fit using model.fit passing in the train data X_train and y_train. See Fig 5.

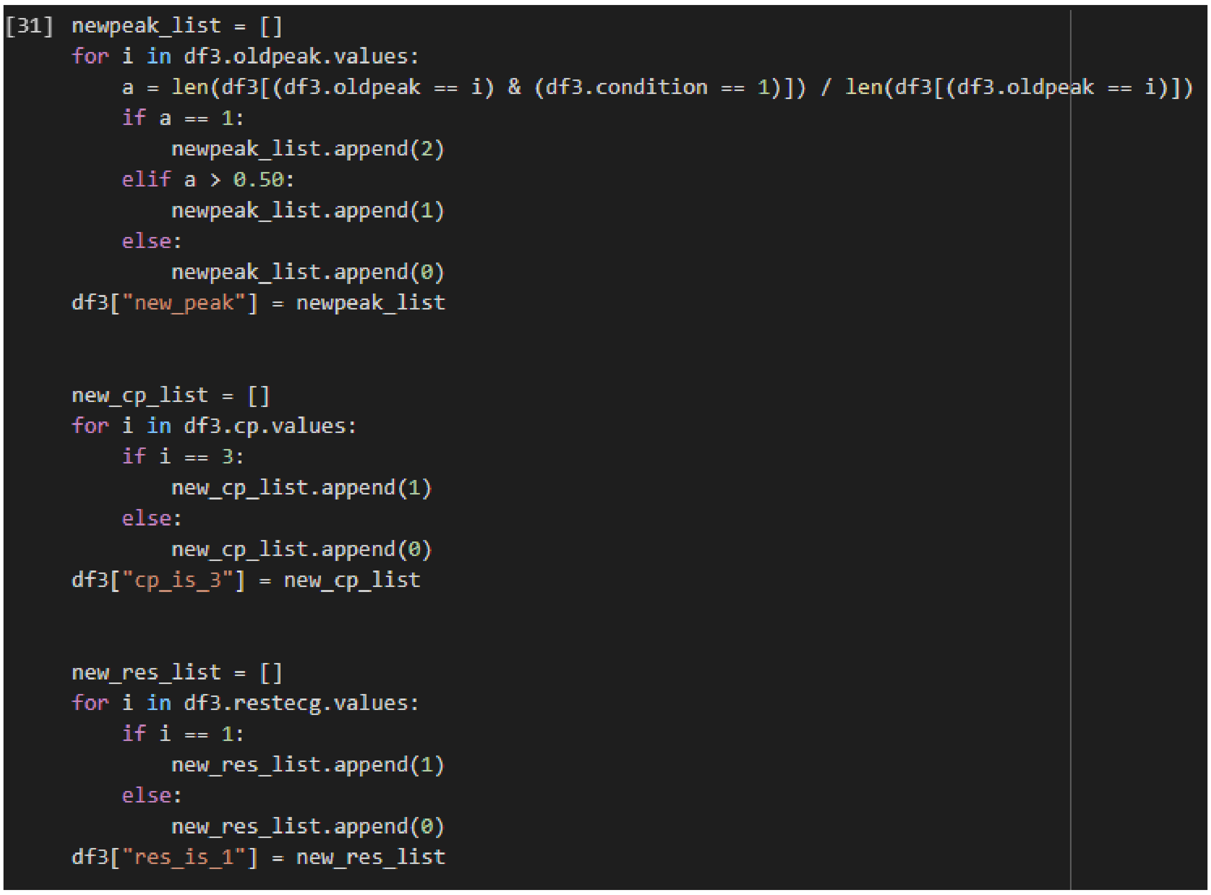


```
#set up model
model = LogisticRegressionCV()

# separating the dependent variables and the independent variables
X = df.iloc[:,:13]
y = df["condition"]
print(X.shape, y.shape)
X.head()


#train test split
(X_train, X_test, y_train, y_test) = train_test_split(X, y, test_size=.2)
```

Fig 5: Declare model and variables.

Once fitted the models accuracy can be calculated using the helper function created earlier. Pass either the test data or training data to see how the model performed and the evaluation metrics. Graphs can also be created visualise the Precision-Recall vs threshold, PR curve and the ROC Curve using each of their respective functions.

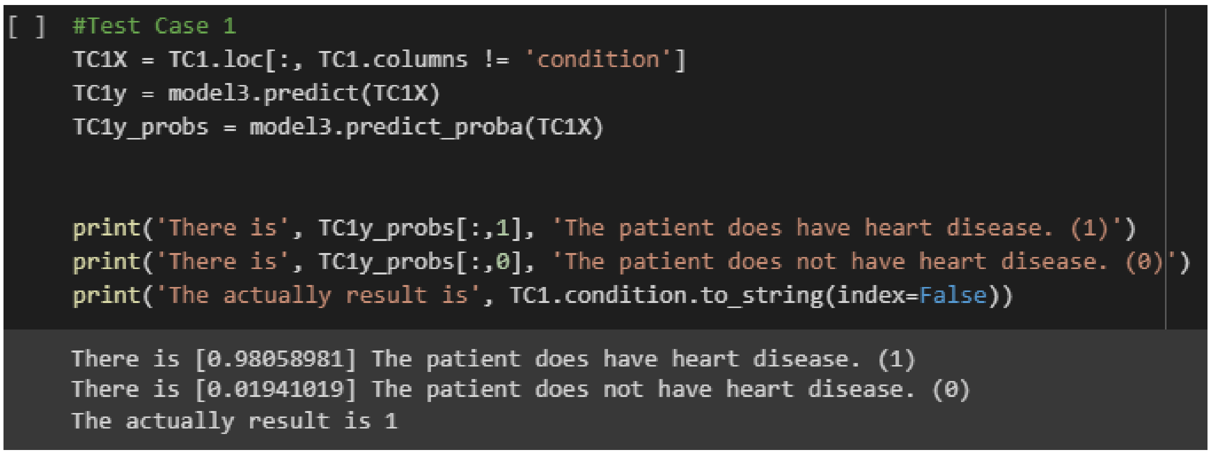
The second model is based on analysis of the correlation matrix created with corrwith() and the target variable. See results section for the matrix data. The matrix shows how each of the features contribute the to the target variable. The goal is to test how the model performs without the features that contribute the least. These columns were dropped and a new model was trained and tested like before without these features. The third model changes the features 'oldpeak', 'cp', 'restecg', 'slope', 'ca' and 'thal' by narrow the data in these columns into categories rather than numbers. For example 'oldpeak' is split into 100%, 50%-99% and >50%. Below is how each some features are altered (Fig 6). The Model then repeated the steps followed in the other models.



```
[31] newpeak_list = []
     for i in df3.oldpeak.values:
         a = len(df3[(df3.oldpeak == i) & (df3.condition == 1)]) / len(df3[(df3.oldpeak == i)])
         if a == 1:
             newpeak_list.append(2)
         elif a > 0.50:
             newpeak_list.append(1)
         else:
             newpeak_list.append(0)
     df3["new_peak"] = newpeak_list


     new_cp_list = []
     for i in df3.cp.values:
         if i == 3:
             new_cp_list.append(1)
         else:
             new_cp_list.append(0)
     df3["cp_is_3"] = new_cp_list


     new_res_list = []
     for i in df3.restecg.values:
         if i == 1:
             new_res_list.append(1)
         else:
             new_res_list.append(0)
     df3["res_is_1"] = new_res_list
```

Fig 6: Example grouping features.

To test the model further we can get the model to predict the target variable from the three test cases extracted from the Cleveland dataset. Model 3 provided the best results when testing so we need to drop the ‘chol’ and ‘fbs’ columns and then run each case through the grouping process. Once completed each test case requires three steps. Firstly create the X and y variables which are the features for X and using the predict function with the X variables for y. Create another variable called TC_prob which will calculate the probability of the test case using. Once the variables are set the output can display the probability of the target variable using TC_prob[:,0] for absence and TC_prob[0:1] for presence. See Fig 6 for test case example.



```
[ ]  #Test Case 1
     TC1X = TC1.loc[:, TC1.columns != 'condition']
     TC1y = model3.predict(TC1X)
     TC1y_probs = model3.predict_proba(TC1X)


     print('There is', TC1y_probs[:,1], 'The patient does have heart disease. (1)')
     print('There is', TC1y_probs[:,0], 'The patient does not have heart disease. (0)')
     print('The actually result is', TC1.condition.to_string(index=False))

     There is [0.98058981] The patient does have heart disease. (1)
     There is [0.01941019] The patient does not have heart disease. (0)
     The actually result is 1
```

Fig 7: Test Case Example

Results

Each Model produces evaluation metrics consisting of accuracy, precision, recall, f1 score and a confusion matrix. See below for descriptions of each metric.

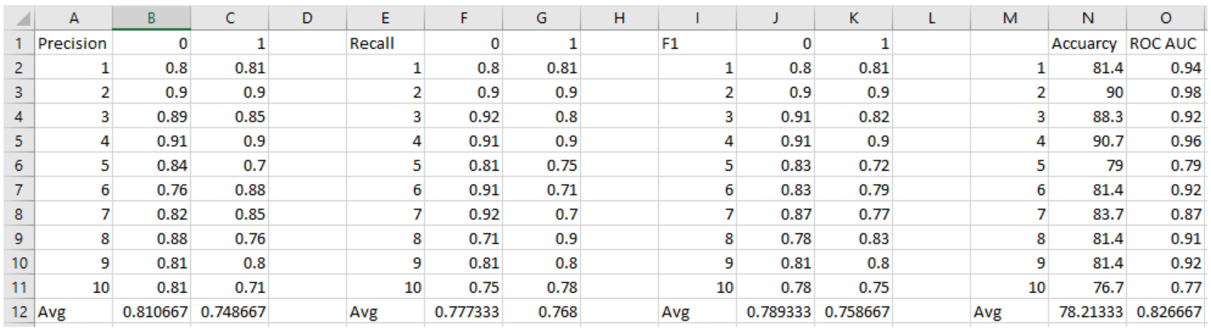
Accuracy: Percentage of correct predictions.

Precision: Number of correct positive prediction / total positive predictions.

Recall: Number of positive predictions / number of positives in the dataset.

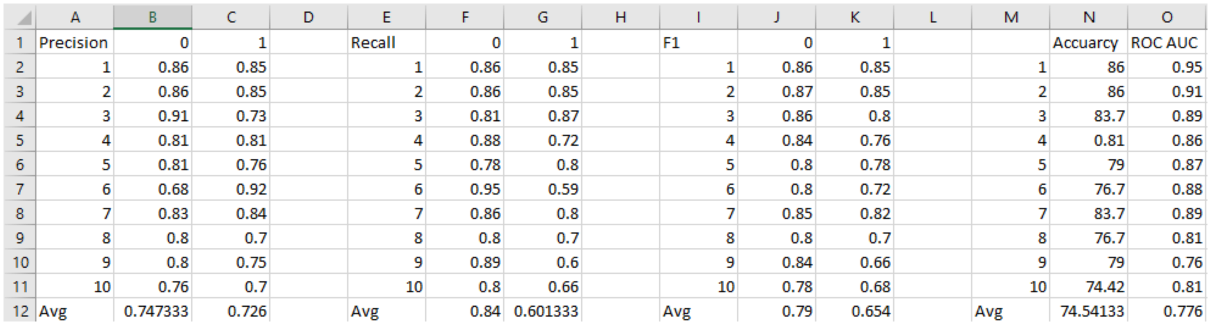
F1 Score: combination of Precision and Recall.

The models also produce graphs of: Precision Recall vs threshold, PR Curve and ROC AUC. The ROC AUC is the Receiver Operator characteristics Area Under Curve. The ROC AUC shows a dotted line to represent the curve of a random classifier and a blue line to show our classifier. The goal is to have the blue line as far from the dotted line as possible and a score as close to 1 as possible. To test the model it was ran ten times, each run the results from the print_score function was extracted into a spreadsheet and calculated into an average score to compare against the other models. See Fig 8-11 for each models results and the calculated averages. Results can also be found in Modelresults.xlsx.



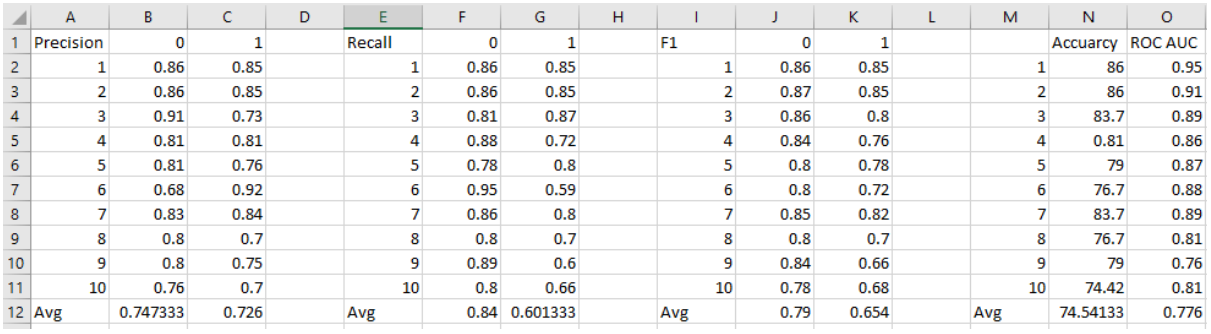
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Precision	0	1		Recall	0	1		F1	0	1			Accuarcy	ROC AUC
2	1	0.8	0.81		1	0.8	0.81		1	0.8	0.81		1	81.4	0.94
3	2	0.9	0.9		2	0.9	0.9		2	0.9	0.9		2	90	0.98
4	3	0.89	0.85		3	0.92	0.8		3	0.91	0.82		3	88.3	0.92
5	4	0.91	0.9		4	0.91	0.9		4	0.91	0.9		4	90.7	0.96
6	5	0.84	0.7		5	0.81	0.75		5	0.83	0.72		5	79	0.79
7	6	0.76	0.88		6	0.91	0.71		6	0.83	0.79		6	81.4	0.92
8	7	0.82	0.85		7	0.92	0.7		7	0.87	0.77		7	83.7	0.87
9	8	0.88	0.76		8	0.71	0.9		8	0.78	0.83		8	81.4	0.91
10	9	0.81	0.8		9	0.81	0.8		9	0.81	0.8		9	81.4	0.92
11	10	0.81	0.71		10	0.75	0.78		10	0.78	0.75		10	76.7	0.77
12	Avg	0.810667	0.748667		Avg	0.777333	0.768		Avg	0.789333	0.758667		Avg	78.21333	0.826667

Fig 8: Model 1 results.



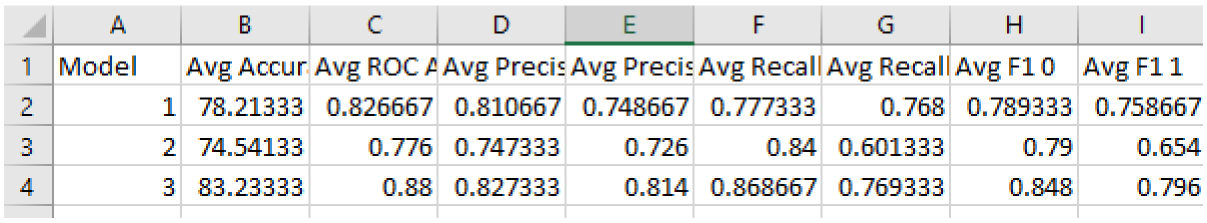
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Precision	0	1		Recall	0	1		F1	0	1			Accuarcy	ROC AUC
2	1	0.86	0.85		1	0.86	0.85		1	0.86	0.85		1	86	0.95
3	2	0.86	0.85		2	0.86	0.85		2	0.87	0.85		2	86	0.91
4	3	0.91	0.73		3	0.81	0.87		3	0.86	0.8		3	83.7	0.89
5	4	0.81	0.81		4	0.88	0.72		4	0.84	0.76		4	0.81	0.86
6	5	0.81	0.76		5	0.78	0.8		5	0.8	0.78		5	79	0.87
7	6	0.68	0.92		6	0.95	0.59		6	0.8	0.72		6	76.7	0.88
8	7	0.83	0.84		7	0.86	0.8		7	0.85	0.82		7	83.7	0.89
9	8	0.8	0.7		8	0.8	0.7		8	0.8	0.7		8	76.7	0.81
10	9	0.8	0.75		9	0.89	0.6		9	0.84	0.66		9	79	0.76
11	10	0.76	0.7		10	0.8	0.66		10	0.78	0.68		10	74.42	0.81
12	Avg	0.747333	0.726		Avg	0.84	0.601333		Avg	0.79	0.654		Avg	74.54133	0.776

Fig 9: Model 2 Results



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Precision	0	1		Recall	0	1		F1	0	1			Accuarcy	ROC AUC
2	1	0.86	0.85		1	0.86	0.85		1	0.86	0.85		1	86	0.95
3	2	0.86	0.85		2	0.86	0.85		2	0.87	0.85		2	86	0.91
4	3	0.91	0.73		3	0.81	0.87		3	0.86	0.8		3	83.7	0.89
5	4	0.81	0.81		4	0.88	0.72		4	0.84	0.76		4	0.81	0.86
6	5	0.81	0.76		5	0.78	0.8		5	0.8	0.78		5	79	0.87
7	6	0.68	0.92		6	0.95	0.59		6	0.8	0.72		6	76.7	0.88
8	7	0.83	0.84		7	0.86	0.8		7	0.85	0.82		7	83.7	0.89
9	8	0.8	0.7		8	0.8	0.7		8	0.8	0.7		8	76.7	0.81
10	9	0.8	0.75		9	0.89	0.6		9	0.84	0.66		9	79	0.76
11	10	0.76	0.7		10	0.8	0.66		10	0.78	0.68		10	74.42	0.81
12	Avg	0.747333	0.726		Avg	0.84	0.601333		Avg	0.79	0.654		Avg	74.54133	0.776

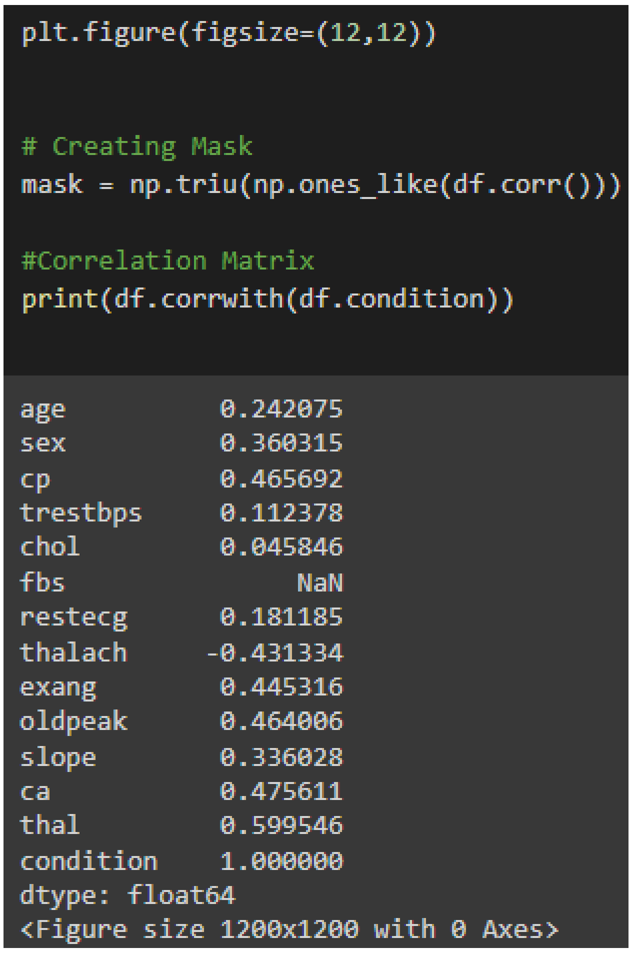
Fig 10: Model 3 Results



	A	B	C	D	E	F	G	H	I
1	Model	Avg Accur	Avg ROC A	Avg Precis	Avg Precis	Avg Recall	Avg Recall	Avg F1 0	Avg F1 1
2	1	78.21333	0.826667	0.810667	0.748667	0.777333	0.768	0.789333	0.758667
3	2	74.54133	0.776	0.747333	0.726	0.84	0.601333	0.79	0.654
4	3	83.23333	0.88	0.827333	0.814	0.868667	0.769333	0.848	0.796

Fig 11: Results comparison

Other data that was collected was a correlation matrix. This was done to see how each feature contributes to the target variable. The matrix can be produced with the corrwith() passing in the target variable this being 'condition'. See Fig 12 for correlation matrix and code.



```
plt.figure(figsize=(12,12))


# Creating Mask
mask = np.triu(np.ones_like(df.corr()))

#Correlation Matrix
print(df.corrwith(df.condition))
```

```
age          0.242075
sex          0.360315
cp           0.465692
trestbps     0.112378
chol         0.045846
fbs               NaN
restecg      0.181185
thalach     -0.431334
exang        0.445316
oldpeak      0.464006
slope        0.336028
ca           0.475611
thal         0.599546
condition    1.000000
dtype: float64
<Figure size 1200x1200 with 0 Axes>
```

Fig 12: Correlation matrix

The matrix was used to identify features from model 1 that could be dropped due to the low correlation. Analysis of the matrix concluded that 'fbs' and 'chol' corelate the least to the target and were removed for subsequent models.

Confusion Matrixes are also produced by the print_score function. This indicates how many true positive, true negative, false positive and false negatives were calculated. The aim is to have as many true positives and true negatives as possible however if the model does make an incorrect prediction it would preferably a false positive this way a patients diagnosis does not go unnoticed. See Fig 12 for example confusion matrix.

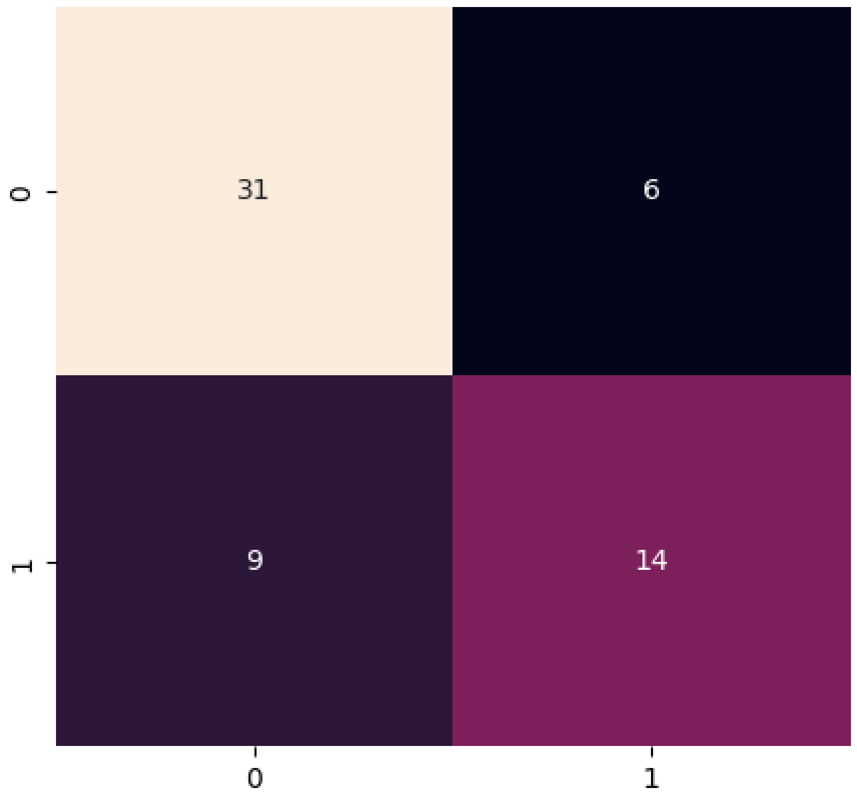


Fig 13: Confusion matrix.

Discussion and Conclusion

The goal of the project is to create a model that can be used to determine a patients risk of heart disease. The model should not be considered a final diagnosis but should determine the risk and then advise the patient to receive further information from a doctor. When evaluating results recall is the main metric to aim for. For this model high recall is necessary as any positive result should be further examined by a doctor.

From each models results we can see that model 1 produced good results which would have hopefully improved with feature analysis from the correlation matrix. This proved not to be the case as when comparing model 1 to model 2, model 2 scored lower accuracy, ROC AUC and overall recall. Model 3 however is where there was an improvement over both model 2 and model 1. It produces the highest values in all metrics with noticeable improvements such as 5% higher accuracy and 5% higher overall recall.

The model that should be used as final is model 3. The grouping method used comes from the visualizations shown in the feature histogram. The features that go through the grouping method can be narrowed down from their original format to a more concise version. For example the 'ca' feature has four values 0, 1, 2 and 3 however 0 makes up more than half of the total values. See Fig 14.

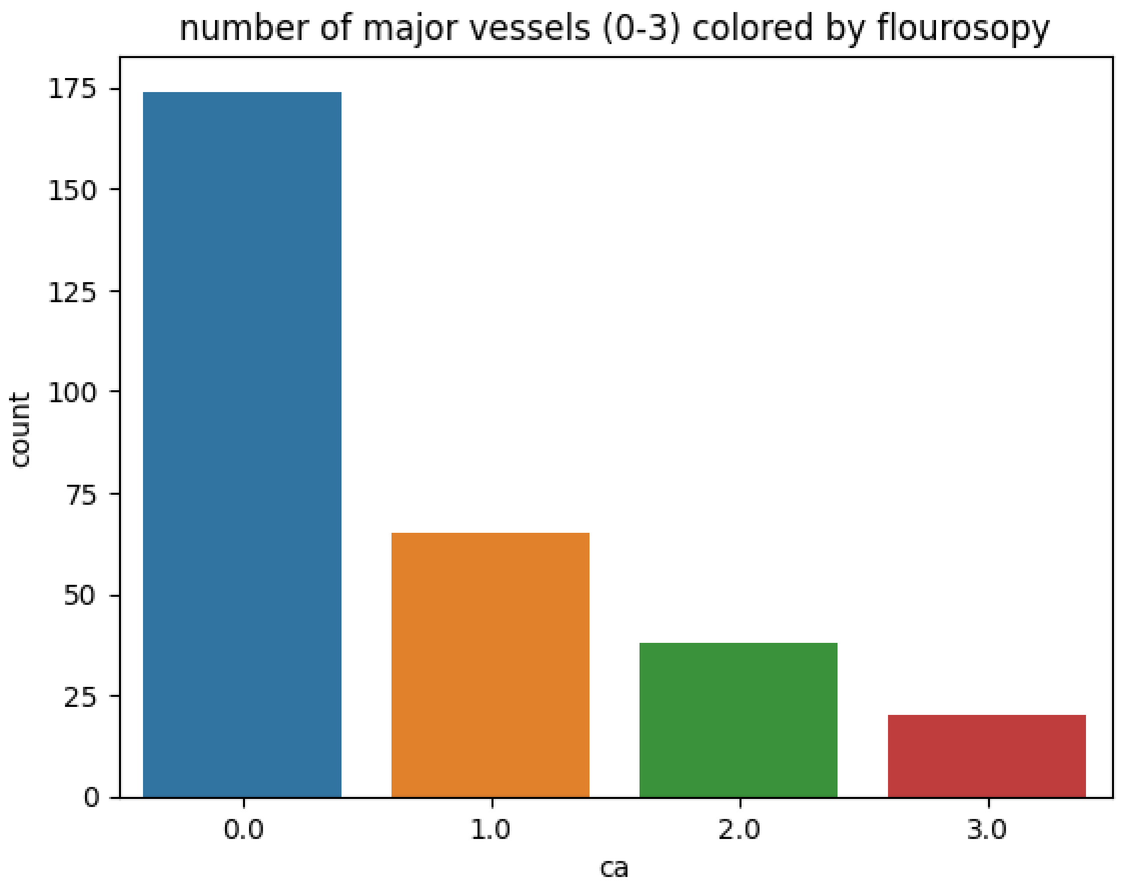


Fig 14: ‘ca’ Histogram

The grouping method turns the data from four values down to just two, 0 or not 0. This is called cardinality which is the number of unique values that a feature has. High cardinality makes it difficult for the model to identify patterns in the dataset (Sangani, 2021). This is most noticeable in the ‘oldpeak’ feature with 40 unique values.

The main limitation faced when creating and testing the model was the data. The dataset only contains 303 datapoints to begin with and decreases after pre-processing. With more patient data the model could be trained further to aim for improved metrics. This can also run the risk of overfitting and a balance must be struck between to much data and too little. To answer this situation a learning curve graph can be used to see where the training drops off when the curve levels off.

Overall three models were trained to determine a patients risk of heart disease which started with an average 78% accuracy and 0.82 ROC AUC to a final improve model of 83% and 0.88 by using an improved feature grouping function. This model can also be further improved by the addition of more patient data up to the learning curve.

References

British Heart Foundation (2023) *Heart statistics, British Heart Foundation*. Available at: https://www.bhf.org.uk/what-we-do/our-research/heart-statistics (Accessed: 03 May 2023).

Marangoz, B. (2023) *Logistic Regression - ROC: %96 , ACC: %90*. Available at: https://www.kaggle.com/code/burakmarangoz88/logistic-regression-roc-96-acc-90 (Accessed: 03 May 2023).

Detrano, R. (1998) 'Heart Disease Data Set'. Cleveland. Available at: https://archive.ics.uci.edu/ml/datasets/Heart+Disease (Accessed: 03 May 2023).

IBM (no date) *What is logistic regression?, IBM*. Available at: https://www.ibm.com/topics/logistic-regression (Accessed: 28 April 2023).

Sayah, F. (2023) *Logistic Regression for Binary Classification Task*. Available at: https://www.kaggle.com/code/faressayah/logistic-regression-for-binary-classification-task#2.-Theory-Behind-Logistic-Regression (Accessed: May 4, 2023).

Team, G.L. (2022) *What is Cross Validation in machine learning? types of cross validation, Great Learning Blog: Free Resources what Matters to shape your Career!* Available at: https://www.mygreatlearning.com/blog/cross-validation/#:~:text=The%20purpose%20of%20cross–validation,generalize%20to%20an%20independent%20dataset. (Accessed: May 4, 2023).

Sangani, R. (2021) *Dealing with features that have high cardinality, Medium*. Available at: https://towardsdatascience.com/dealing-with-features-that-have-high-cardinality-1c9212d7ff1b#:~:text=In%20most%20cases%2C%20high%20cardinality,examples%20outside%20the%20training%20set. (Accessed: 09 May 2023).