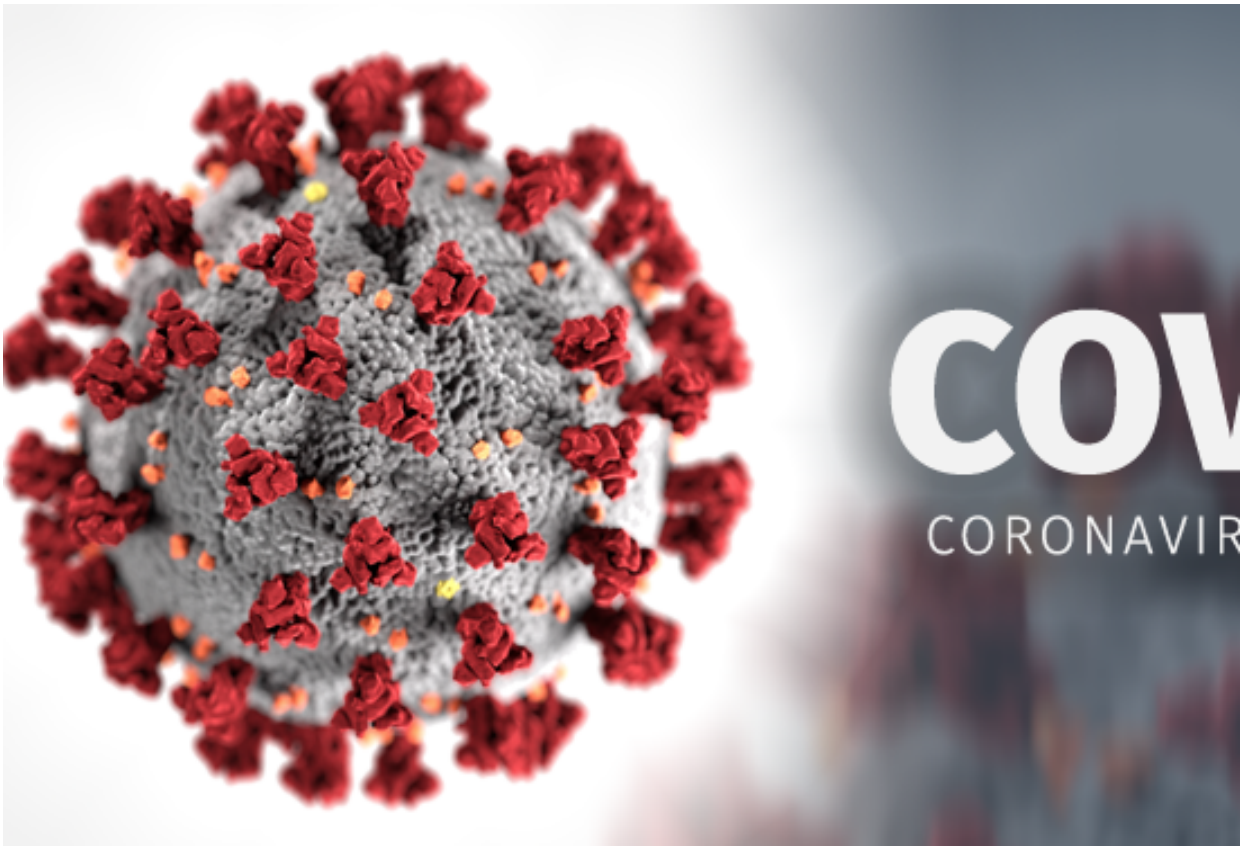
In[8]:=

Mathematica Research Project

Kevin O'
Sullivan



Project Introduction

Please note that due to re sampling / sub sampling of the original dataset which is too large for my computer, there may be inconsistencies in some of the output cells and the descriptions.

The main focus of this project is to implement various pieces of the mathematica toolset, in particular some of the parts of mathematica covered in the ACM40730 module “Introduction to Mathematica”. The primary goal of the project is to develop a mortality predictor for individuals who have been infected by the viral infection Covid-19. This viral infection has taken the world by surprise and put pressure on medical resources. For this reason it is important that a hospital or any medical resource provision centre would be able to classify incoming patients in terms of their likelihood of death. This can allow for the most vulnerable patients to receive the attention and resources

they get. This concept is known as a triage which is defined by Wikipedia to be :

“Triage is the process of determining the priority of patients’ treatments by the severity of their condition or likelihood of recovery with and without treatment.”

There are a number of sections in this project and they are outlined as follows :

1. Dataset Description
2. Data pre processing
3. Exploratory Data Analysis
4. Model Training and Evaluation
5. Visualisations
6. Conclusions

Dataset Description

The dataset is available in a github repository here : https://github.com/beoutbreakprepared/nCoV2019
The dataset is described in detail in this research paper: https://rdcu.be/cbZzD

The dataset was curated by a group of researchers Xu, B., Gutierrez, B., Mekaru, S. et al.. It is a conglomerate of datasets
which were made available by government bodies and various groups. Patient level covid 19 data is very limited at this point in time,
with most current covid-19 data being used for aggregated analysis such as cases per region. This dataset contains 26 million records.
Many of the columns however only have a very small few consistent entries making them not appropriate for model training.

Data Pre-Processing

Data Import

In[9]:=
```
file = Import[
   "/Users/kevinosullivan/Desktop/Masters/Semester 1/Mathematica/Covid-19/sub.
     csv"];
headers = file[[1]];
data = file[[2 ;;]];
```

In[12]:=
```
testset = Thread[headers → #] & /@ data // Map[Association] // Dataset;
```

The data set has been imported. The dataset has approximately 2.6 million records. The columns in the data are :

In[13]:=
```
testset[1, Keys]
```

Out[13]=

ID	age	sex
city	province	countr
latitude	longitude	geo_re
date_onset_symptoms	date_admission_hospital	date_c
symptoms	lives_in_Wuhan	travel_
travel_history_location	reported_market_exposure	additi
chronic_disease_binary	chronic_disease	source
sequence_available	outcome	date_c
notes_for_discussion	location	admin
admin2	admin1	countr
admin_id	data_moderator_initials	travel_

Data Selection

The dataset contains a large number columns, however , many of the columns will not be needed for this project. The columns
needed for this project will be any values that could be useful for mortality prediction as well as some columns for some
exploratory analysis and visualisations.

Many of the columns have very few values eg. chronic_disease has only 143 examples with values in the 2.6 million patients.
While this would be useful information there is very limited data here to train a model. For this reason the variables used in
the model are limited. A model will be trained with only the variables age, sex, chronic_disease_binary and outcome as the
dependent variable which are the variables with the most consistent examples and have a sample of 33,539 patients. There
is very limited availability of patient level covid 19 data at this point.

Number of values for chronic_disease in dataset :

In[14]:=
```
Length[testset[Select[#"chronic_disease" ≠ "" &]]]
```

Out[14]= 138

The variables in the dataset I will be using for the mortality prediction model are :

- age : the age of the patient
- sex : the sex of the patient
- chronic_disease_binary : a binary variable describing whether a patient has a chronic disease or not

- outcome : describes the outcome of the patient eg.recovered

In[15]:=
```
modelData =
 testset[Select[NumericQ[#age] && #sex ≠ "" && #"chronic_disease_binary" ≠ "" &&
     #"outcome" ≠ "" &], {"age", "sex", "chronic_disease_binary", "outcome"}]
```

Out[15]=

age	sex	chronic_disease_binary	outcome
78	male	FALSE	death
61	female	FALSE	discharge
28	male	FALSE	discharge
56	female	FALSE	discharge
79	female	FALSE	discharge
26	male	FALSE	discharge
25	male	FALSE	discharge
40	male	FALSE	discharge
43	male	FALSE	discharge
29	male	FALSE	discharge
71	female	FALSE	discharge
68	female	FALSE	death
1	male	FALSE	discharge
35	male	FALSE	discharge
36	male	FALSE	discharge
32	male	FALSE	discharge
30	female	FALSE	discharge
41	male	FALSE	discharge
58	female	FALSE	discharge
38	male	FALSE	discharge

rows 1–20 of 2258

Data Manipulation & Pre Processing

Some of the columns contain categorical variables and these will have to be split into levels and represented by binary variables.

Convert variable sex to binary representation (0 = female | 1 = male)

In[16]:=
```
modelData =
  modelData[All, If[#"sex" == "male", <|#, "sex" → 1|>, <|#, "sex" → 0|>] &];
```

Convert variable chronic_disease_binary to binary representation (0 = No Chronic Disease | 1 = Chronic Disease)

In[17]:=
```
modelData = modelData[All, If[#"chronic_disease_binary" == "True", <|#,
      "chronic_disease_binary" → 1|>, <|#, "chronic_disease_binary" → 0|>] &];
```

Convert the values of outcome column to either 1 (Dead) or 0 (Recovered). Some values in this column are not useful here such as “hospitalised” or “critical condition”.
Thees are not potential outcomes that our classifier will consider as we are predicting mortality, which is a binary value.

In[18]:=
```
modelData = modelData[All,
   If[#"outcome" == "death" || #"outcome" == "Dead" || #"outcome" == "Death" ||
      #"outcome" == "Died" || #"outcome" == "dead" || #"outcome" == "died" ||
      #"outcome" == "Deceased", <|#, "outcome" → 1|>, <|
      #, "outcome" → #outcome|>] &];
modelData = modelData[All, If[#"outcome" == "discharge" ||
      #"outcome" == "discharged" || #"outcome" == "Discharged" ||
      #"outcome" == "recovered" || #"outcome" == "Alive" || #"outcome" ==
       "Recovered", <|#, "outcome" → 0|>, <|#, "outcome" → #outcome|>] &];
modelData = modelData[Select[#outcome == 0 || #outcome == 1 &], All];
```

Exploratory Data Analysis

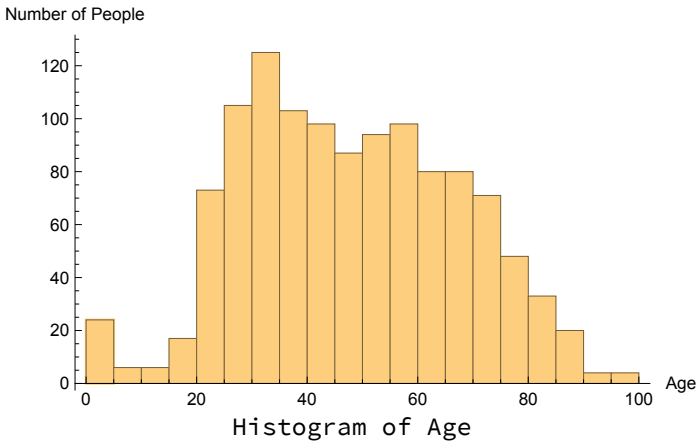
In this section I will explore the data. I will look at the distributions of the different variables and also the relationships between the relevant variables.

Age

Histogram of age:

In[21]:=
```
Labeled[Histogram[modelData[All, "age"],
  AxesLabel → {"Age", "Number of People"}], "Histogram of Age"]
```

Out[21]=



This shows the distribution of the age variable. There are very few young people in this dataset, The age groups of 0-5, 5-10, 10-15 and15-20 years old all have few examples in the dataset.
Most of the dataset is between 20 and 80 years old. There are very few examples older than 80 also. This is to be expected. The 30-35 years old bar is highest and this agegroup has more indiviuals than any other agegroup.

In[22]:=
```
Median[modelData[All, "age"]]
```

Out[22]= 46

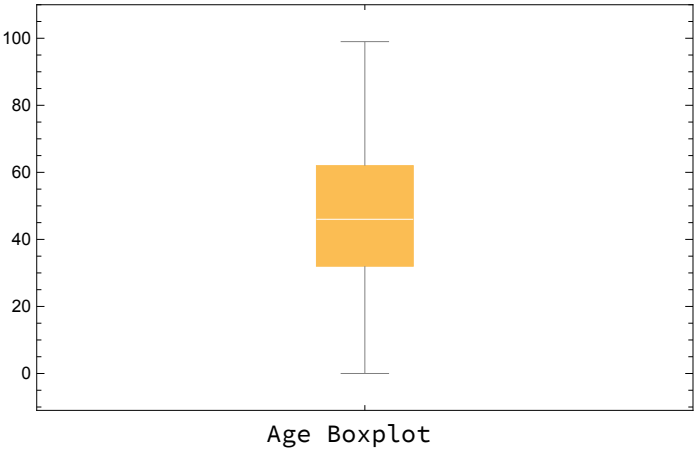
In[23]:=
```
Mean[modelData[All, "age"]]
```

Out[23]= 47.341

The median value for age is 46 and the mean is very similar at 47.0782.

In[24]:=
```
Labeled[BoxWhiskerChart[modelData[All, "age"]], "Age Boxplot"]
```

Out[24]=

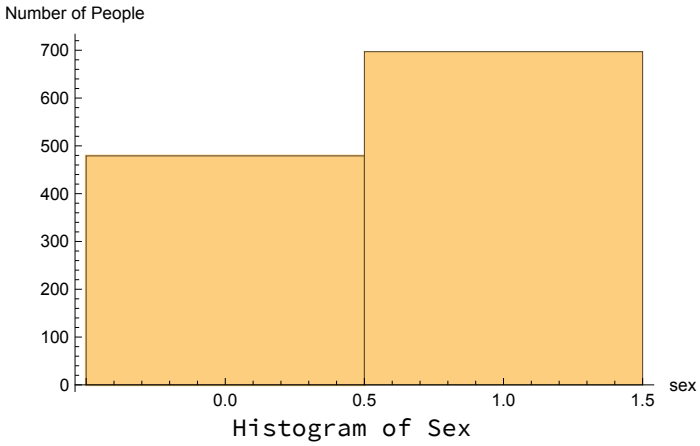


The box plot shows the distribution of the age variable. The interquartile range goes from 32 to 61. So 25% of the dat has age below 32 and 75% has age below 61.

Sex

In[25]:=
```
Labeled[Histogram[modelData[All, "sex"],
  AxesLabel → {"sex", "Number of People"}], "Histogram of Sex"]
```

Out[25]=



The histogram shows the two groups 0 = female and 1 = male . There are slightly more males than females in the dataset.

In[26]:=
```
Labeled[GroupBy[modelData, "sex", Length], "Frequency Table for sex"]
```

Out[26]=

1	697
0	479

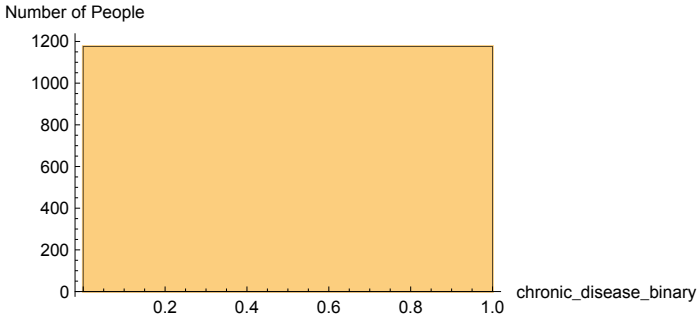
Frequency Table for sex

There are 2912 males and 2491 females in the dataset.

Chronic Disease Binary

In[27]:=
```
Labeled[Histogram[modelData[All, "chronic_disease_binary"],
  AxesLabel → {"chronic_disease_binary", "Number of People"}],
 "Histogram of Chronic Disease Binary"]
```

Out[27]=



Histogram of Chronic Disease Binary

Group with chronic_disease_binary=0 do not have a chronic disease, they make up almost all of the dataset. About 5100 of the people have no chronic disease. A small percentage of the data has a chronic disease, about 100 people.

In[28]:=
```
Labeled[GroupBy[modelData, "chronic_disease_binary", Length],
 "Frequency Table for chronic disease binary"]
```

Out[28]=

0	1176

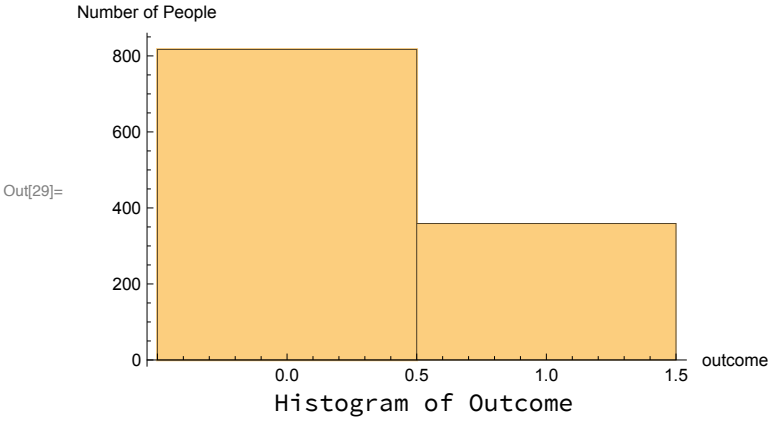
Frequency Table for chronic disease binary

121 of the patients have a chronic disease and 5282 do not.

Outcome

In[29]:=
```
Labeled[Histogram[modelData[All, "outcome"],
  AxesLabel → {"outcome", "Number of People"}], "Histogram of Outcome"]
```

Out[29]=



The histogram shows that significantly more people survived than died.

In[30]:=
```
Labeled[GroupBy[modelData, "outcome", Length], "Frequency Table for outcome"]
```

Out[30]=

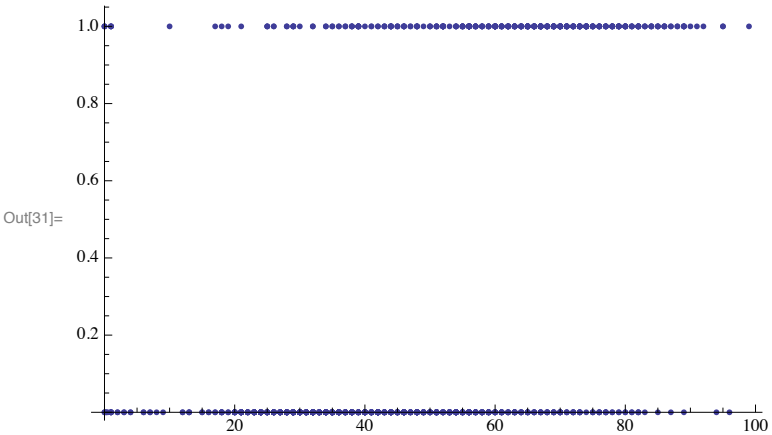
1	359
0	817

Frequency Table for outcome

Age Vs Outcome

In[31]:=
```
ListPlot[
 Transpose[{Normal[modelData[All, "age"]], Normal[modelData[All, "outcome"]]}],
 PlotTheme → "Classic"]
```

Out[31]=



The scatterplot of age vs outcome shows two distinct lines of dots. One line is at outcome = 0 this is the group that survived. The group at outcome = 1 is the group that died.
The group with outcome = 0 seems to be more concentrated to the left than the group outcome = 1. This would suggest a lower expected age in survivors. I will
investigate this with some descriptive statistics in particular the mean and median. These are both

measures of centre.

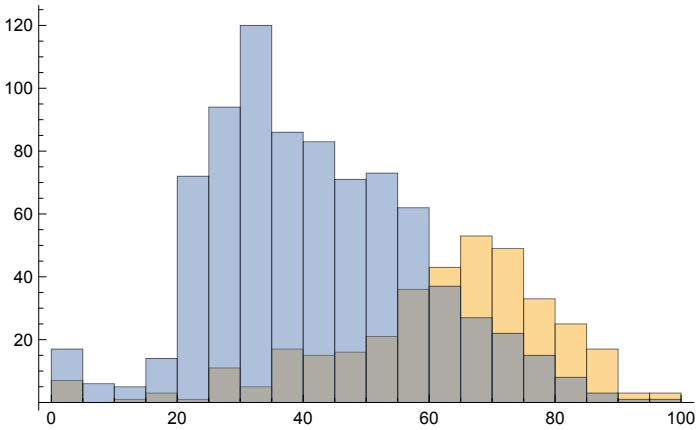
In[32]:=
```
Median[modelData[Select[#"outcome" == 1 &], "age"]]
```
Out[32]= 65

In[33]:=
```
Median[modelData[Select[#"outcome" == 0 &], "age"]]
```
Out[33]= 39

In[34]:=
```
40
```
Out[34]= 40

As expected, the median of the group that died is 64 and is significantly higher than the group that survived, whose average age is 40 years old.

In[35]:=
```
Histogram[{Normal@modelData[Select[#"outcome" == 1 &], "age"],
  Normal@modelData[Select[#"outcome" == 0 &], "age"]}]
```
Out[35]=



The group that died is plotted in orange. It is much smaller and its median value is much higher than the group that recovered. The
group that died is significantly older on average and most of its data is distributed between 40 and 80 years old.

In[36]:=

In[37]:=
```
N[Mean[modelData[Select[#"outcome" == 1 &], "age"]]]
```
Out[37]= 61.493

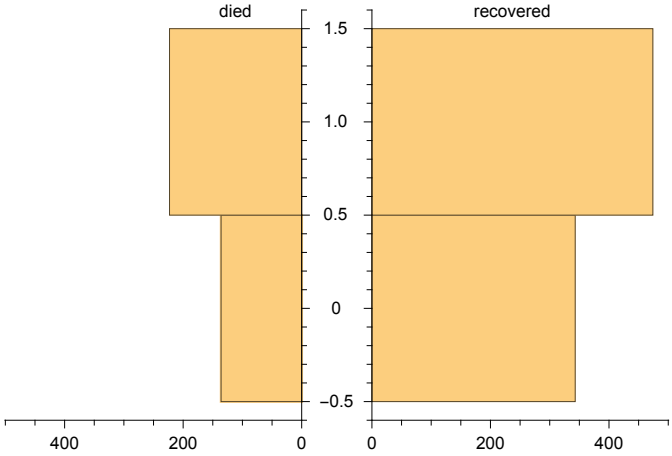
In[38]:=
```
N[Mean[modelData[Select[#"outcome" == 0 &], "age"]]]
```
Out[38]= 41.1224

Similarly, the mean of the group that died is 61.77, and the mean of the group that recovered is 42.3.

Sex Vs Outcome

In[39]:=
```
PairedHistogram[Normal@modelData[Select[#"outcome" == 1 &], "sex"],
 Normal@modelData[Select[#"outcome" == 0 &], "sex"],
 ChartLabels → {"died", "recovered"}]
```

Out[39]=



The group with that recovered has almost equal males and females. The group that died has significantly more males than females

In[40]:=
```
GroupBy[modelData[Select[#"outcome" == 1 &], All], "sex", Length]
```

Out[40]=

1	223
0	136

In[41]:=
```
GroupBy[modelData[Select[#"outcome" == 1 &], All],
 "chronic_disease_binary", Length]
```

Out[41]=

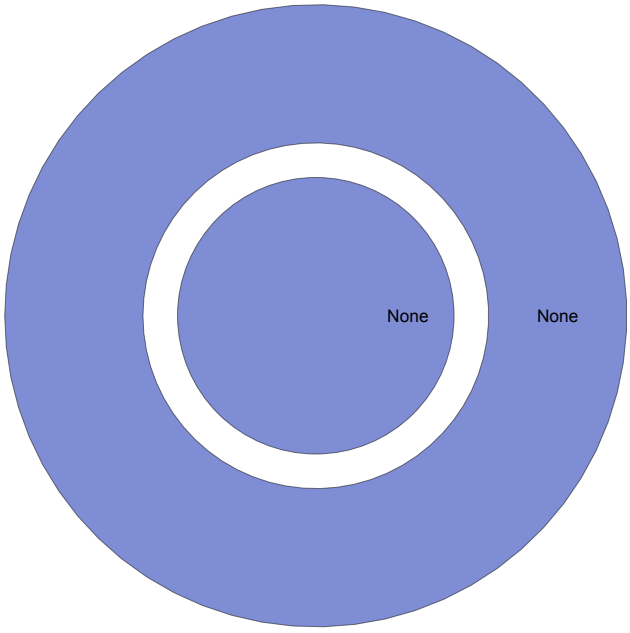
0	359

There are 841 males that died and only 483 females.

Chronic Disease Binary Vs Outcome

In[42]:=
```
PieChart[
 {{Length[modelData[Select[#"outcome" == 1 && #"chronic_disease_binary" == 1 &],
     "chronic_disease_binary"]],
   Length[modelData[Select[#"outcome" == 1 && #"chronic_disease_binary" == 0 &],
     "chronic_disease_binary"]]},
  {Length[modelData[Select[#"outcome" == 0 && #"chronic_disease_binary" == 1 &],
     "chronic_disease_binary"]],
   Length[modelData[Select[#"outcome" == 0 && #"chronic_disease_binary" == 0 &],
     "chronic_disease_binary"]]}},
 ChartLabels → {"Chronic Disease", "None"}]
```

Out[42]=



This donut chart shows the portion of each group that had chronic disease. The central pie chart has the group that died, there
is a significant portion of the group that has a chronic disease. In the group that recovered it is a very small percentage.

In[43]:=

In[44]:=
```
Labeled[PercentForm[
  N[Length[modelData[Select[#"outcome" == 1 && #"chronic_disease_binary" == 1 &],
      "chronic_disease_binary"]] /
    Length[modelData[All, "chronic_disease_binary"]]]],
 "Percentage of chronic disease in Died", Top]
```

```
Percentage of chronic disease in Died
```

Out[44]=

0%

In[45]:=
```
Labeled[PercentForm[
  N[Length[modelData[Select[#"outcome" == 0 && #"chronic_disease_binary" == 1 &],
      "chronic_disease_binary"]] /
    Length[modelData[All, "chronic_disease_binary"]]]],
 "Percentage of chronic disease in Recovered", Top]
```

Percentage of chronic disease in Recovered

Out[45]= 0%

There is 1.869% rate of chronic disease in those that died. There is only 0.3702% in those that recovered.This indicates there is a relationship between the variables.

Model Training and Evaluation

In this section I will train and evaluate a number of models in order to assess which model is the most appropriate.
The models that will be trained and evaluated are : Nearest Neighbour, Logistic Regression, and an Artificial Neural Network

First give each row in the data a unique ID :

In[46]:=
```
modelData = modelData[AssociationThread[Range@Length@#, #] &];
```

In the model, the predictor variables are age, sex and chronic_disease_binary. These will be the features for the machine learning model. The dependent variable is outcome.
The KeyDrop function is used here which drops all elements with this key, in this case outcome, so the other keys are used.

In[47]:=
```
featSplit = modelData[All, <|
    "Features" → Values@*KeyDrop["outcome"], "Objective" → Key["outcome"]|>];
```

I will use an 80/20 split for my training and test data . this means 4322 will be in the training set and 1081 for testing.

In[48]:=
```
splitNum = Round[modelData[Length] 0.2];
ids = Range@modelData[Length];
testIds = ids ~ RandomSample ~ splitNum;
trainIds = ids ~ Complement ~ testIds;
tt = featSplit[<|"Test" → testIds, "Train" → trainIds|>];
trainData = tt["Train"];
testData = tt["Test"];
```

In[55]:=

Nearest Neighbour

The Nearest Neighbours algorithm is used for classification and regression. When classifying it assigns the average value of the k nearest neighbours of the data.

Train the model

In[56]:=
```
knnFun = Classify[Values[trainData, #Features &] ->
    Values[trainData, #Objective &], Method → "NearestNeighbors"];
```

In[57]:=
```
Information[knnFun, "MethodDescription"]
```

Out[57]=

The nearest neighbors classifier infers the class of a new example by analyzing its nearest neighbors in the feature space. In its simplest form, it picks the commonest class amongst the "k"–nearest neighbors.

Append Test Classifications

Now I will use the trained model to classify the test set, and append the classification results as a new column .

In[58]:=
```
featSplit =
  tt[{"Test" → Query[All, Append[#, <|"Classified as" → knnFun@#Features|>] &]}];
```

In[59]:=

Now each example contains:

In[60]:=
```
featSplit["Test", 5]
```

Out[60]=

Features	{80, 0, 0}
Objective	0
Classified as	1

Classifier Evaluation

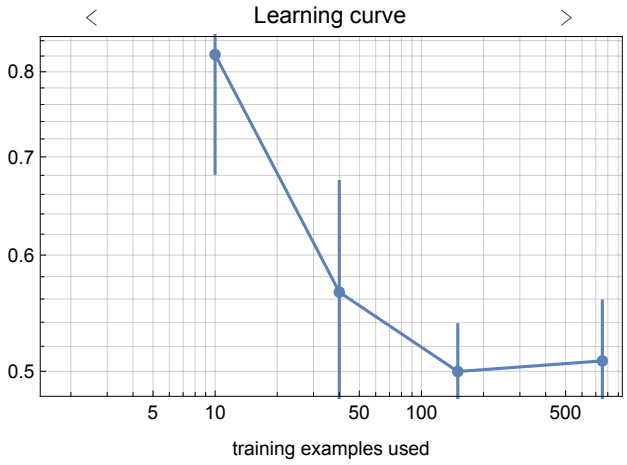
In[61]:=
```
ClassifierInformation[knnFun]
```

... ClassifierInformation: ClassifierInformation is obsolete. It has been superseeded by Information since version 12.

Out[61]=

Classifier information

Data type	{Numerical, Boolean, Numerical}
Classes	0, 1
Accuracy	(79.3 ± 2.4)%
Method	NearestNeighbors
Single evaluation time	2.41 ms/example
Batch evaluation speed	57.4 examples/ms
Loss	0.500 ± 0.038
Model memory	158. kB
Training examples used	941 examples
Training time	567. ms



The model used 4322 examples which is 80% of the data for training. The data types of the model features are
Numerical, Boolean, Boolean. Age is numerical and sex and chronic disease binary are booleans.

In order to evaluate the model I will create a classifier measurements object and access a number of the available classifier measurements.

The classifier measurements I will use here are :

- Accuracy : The classification accuracy of the classifier. This is the fraction of classifications which are correct.
- Error : The error rate of the classifier. Accuracy + Error = 1.
- Precision : This is the proportion of positive classifications which are correct.
- Recall : This is the true positive rate.
- F1 Score : provides an average measure of recall and precision. A measure of models accuracy.

In[62]:=
```
measurementsKNN = ClassifierMeasurements[knnFun,
   Values[testData, #Features &] -> Values[testData, #Objective &]];
knnRes = Labeled[TextGrid[{{"Accuracy", "Error", "Precision",
     "Recall", "F1Score"}, { measurementsKNN["Accuracy"],
     measurementsKNN["Error"], measurementsKNN["Precision"],
     measurementsKNN["Recall"], measurementsKNN["F1Score"]}}, Frame → All],
  "Nearest Neighbour Classifier Measures", Top]
```

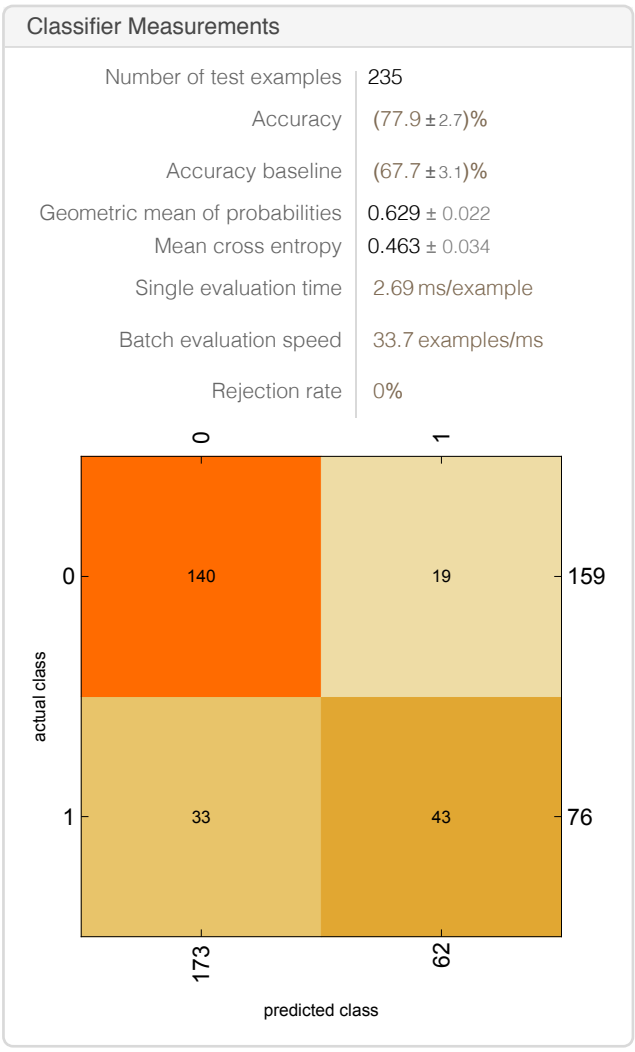
Nearest Neighbour Classifier Measures

Out[63]=

Accuracy	Error	Precision	Recall	F1Score
0.778723	0.221277	⟨|0 → 0.809249, 1 → 0.693548|⟩	⟨|0 → 0.880503, 1 → 0.565789|⟩	⟨|0 → 0.843373, 1 → 0.623188|⟩

In[64]:=
```
measurementsKNN["Report"]
```

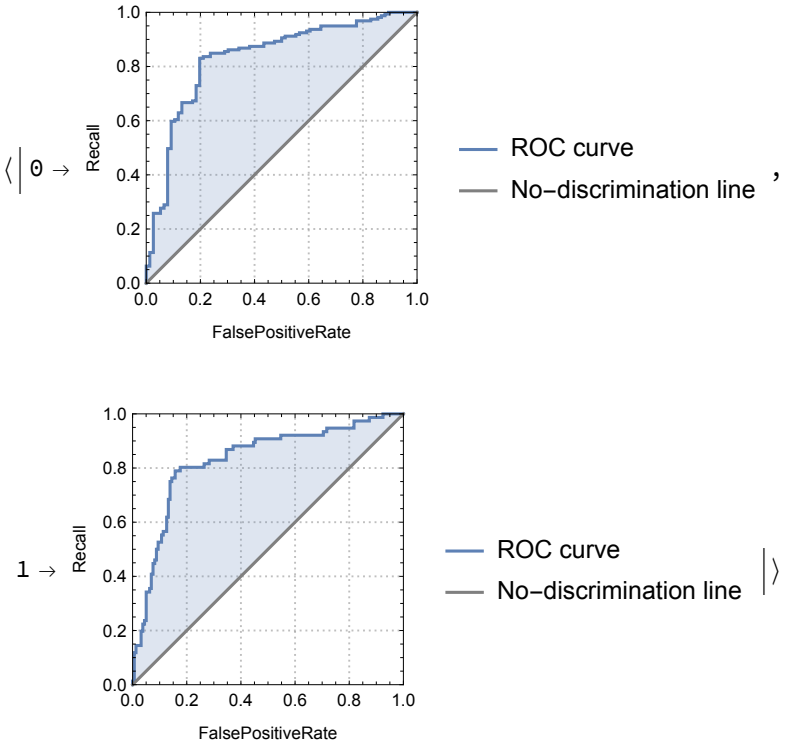
Out[64]=



The confusion matrix shows that 168 of the 271 examples which were of class 1 were predicted as class 0. This might be due to the fact there there is a large imbalance in the data and not many examples of class 1.

In[65]:=
```
measurementsKNN["ROCCurve"]
```

Out[65]=



The ROC curve for both classes are strongly up to the top left which indicates good performance. The diagonal line at 45 degrees represent a randomised classification.

Logistic Regression

Logistic regression is a statistical model that uses a logistic function to model a binary dependent variable. Here I will train a logistic regression model and assess its performance

Train the model

In[66]:=
```
lrFun = Classify[Values[trainData, #Features &] ->
    Values[trainData, #Objective &], Method → "LogisticRegression"];
```

In[67]:=
```
Information[lrFun, "MethodDescription"]
```

Out[67]=

The logistic regression classifier models class probabilities with logistic functions of linear combinations of features. It is also called log−linear model, softmax regression, or maximum−entropy classifier.

In[68]:=

Append Test Classifications

Out[68]=
```
Append Classifications Test
```

Now I will use the trained model to classify the test set, and append the classification results as a new column .

In[69]:=
```
featSplit =
  tt[{"Test" → Query[All, Append[#, <|"Classified as" → lrFun@#Features|>] &]}];
```

In[70]:=

Now each example contains:

In[71]:=
```
featSplit["Test", 5]
```

Out[71]=

Features	{80, 0, 0}
Objective	0
Classified as	1

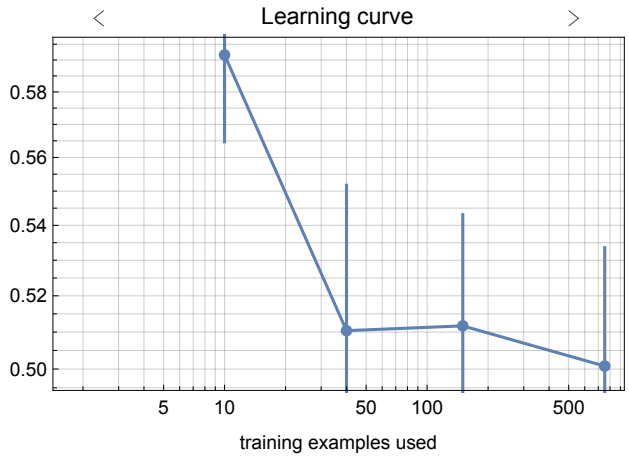
Classifier Evaluation

In[72]:=
```
ClassifierInformation[lrFun]
```

Out[72]=

Classifier information

Data type	{Numerical, Boolean, Numerical}
Classes	0, 1
Accuracy	(79.8 ± 2.8)%
Method	LogisticRegression
Single evaluation time	2.14 ms/example
Batch evaluation speed	176. examples/ms
Loss	0.501 ± 0.033
Model memory	258. kB
Training examples used	941 examples
Training time	1.42 s



The model used 4322 examples which is 80% of the data for training. The data types of the model features are
Numerical, Boolean, Boolean. Age is numerical and sex and chronic disease binary are booleans.

In order to evaluate the model I will create a classifier measurements object and access a number of the available classifier measurements.

The classifier measurements I will use here are :

- Accuracy : The classification accuracy of the classifier. This is the fraction of classifications which are correct.
- Error : The error rate of the classifier. Accuracy + Error = 1.
- Precision : This is the proportion of positive classifications which are correct.
- Recall : This is the true positive rate.
- F1 Score : provides an average measure of recall and precision. A measure of models accuracy.

In[73]:=
```
measurementslr = ClassifierMeasurements[lrFun,
   Values[testData, #Features &] -> Values[testData, #Objective &]];
lrRes = Labeled[TextGrid[{{"Accuracy", "Error", "Precision",
     "Recall", "F1Score"}, { measurementslr["Accuracy"],
     measurementslr["Error"], measurementslr["Precision"],
     measurementslr["Recall"], measurementslr["F1Score"]}},
   Frame → All], "Logistic Regression Measures", Top]
```

Logistic Regression Measures

Out[74]=

Accuracy	Error	Precision	Recall	F1Score
0.782979	0.217021	⟨| 0 → 0.803371, 1 → 0.719298 |⟩	⟨| 0 → 0.899371, 1 → 0.539474 |⟩	⟨| 0 → 0.848665, 1 → 0.616541 |⟩

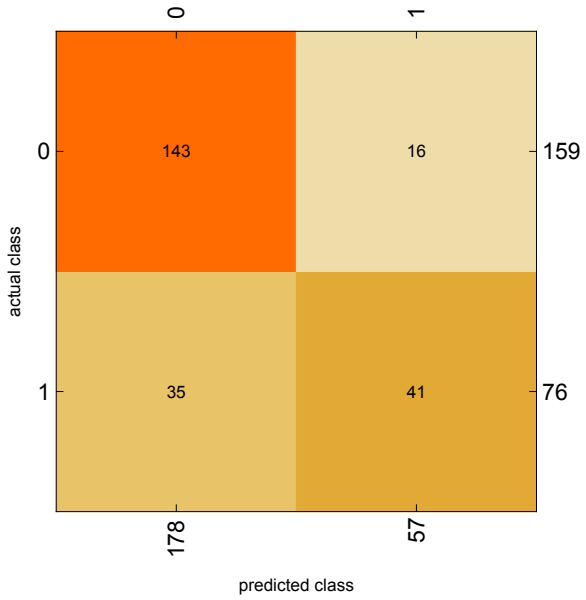
In[75]:=

In[76]:=
```
measurementslr["Report"]
```

Out[76]=

Classifier Measurements

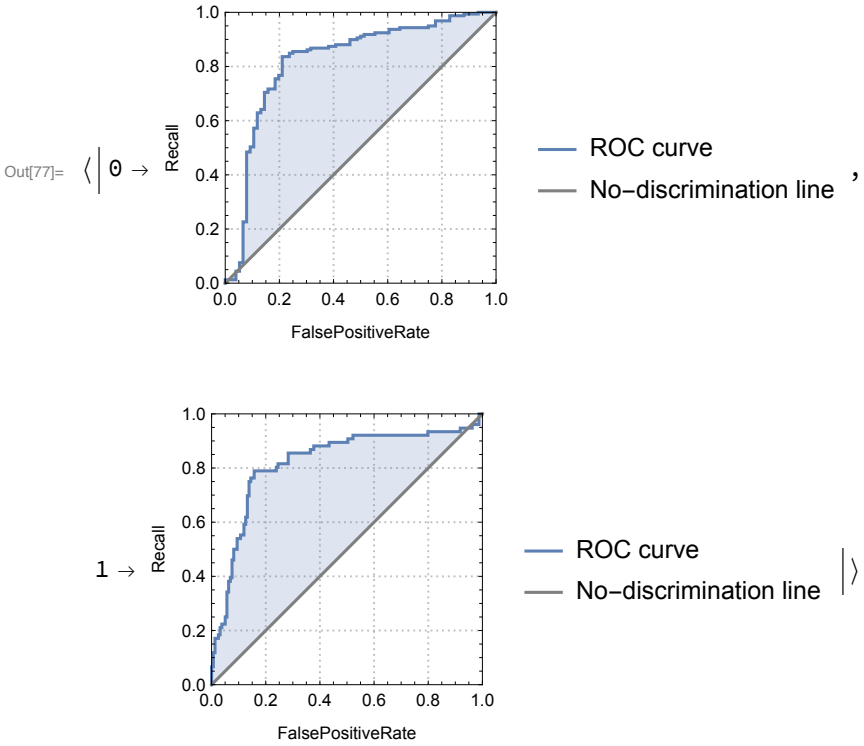
Number of test examples	235
Accuracy	(78.3 ± 2.7)%
Accuracy baseline	(67.7 ± 3.1)%
Geometric mean of probabilities	0.614 ± 0.022
Mean cross entropy	0.488 ± 0.036
Single evaluation time	2.36 ms/example
Batch evaluation speed	87.7 examples/ms
Rejection rate	0%



The confusion matrix shows that 189 of the 289 examples which were of class 1 were predicted as class 0.

In[77]:=
```
measurementslr["ROCCurve"]
```

Out[77]=



The ROC curve for both classes are strongly up to the top left which indicates good performance. The diagonal line at 45 degrees represents a randomised classification.

Neural Network

A neural network is inspired by the human brain's network of neurons. Layers of nodes are weighted to discover patterns in data and create models.

Train the model

In[78]:=
```
nnFun = Classify[Values[trainData, #Features &] ->
    Values[trainData, #Objective &], Method → "NeuralNetwork"];
```

In[79]:=
```
Information[nnFun, "MethodDescription"]
```

Out[79]=

An neural networks is composed of layers of artificial neuron units. Each unit computes its value as a function of the unit values in the previous layer. Information is processed layer by layer from the feature layer to the output layer which gives the class probabilities. It is also called a feed−forward neural network or a multi−layer perceptron.

Append Test Classifications

Now I will use the trained model to classify the test set, and append the classification results as a new column .

In[80]:=
```
featSplit =
  tt[{"Test" → Query[All, Append[#, <|"Classified as" → nnFun@#Features|>] &]}];
```

In[81]:=

Now each example contains:

In[82]:=
```
featSplit["Test", 5]
```

Out[82]=

Features	{80, 0, 0}
Objective	0
Classified as	1

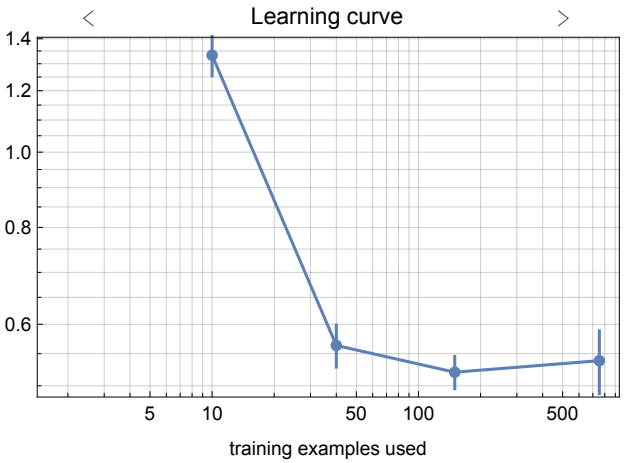
Classifier Evaluation

In[83]:=
```
ClassifierInformation[nnFun]
```

Out[83]=

Classifier information

Data type	{Numerical, Boolean, Numerical}
Classes	0, 1
Accuracy	(77.6 ± 2.1)%
Method	NeuralNetwork
Single evaluation time	3.66 ms/example
Batch evaluation speed	57.8 examples/ms
Loss	0.521 ± 0.025
Model memory	429. kB
Training examples used	941 examples
Training time	14.2 s



The model used 4322 examples which is 80% of the data for training. The data types of the model features are

Numerical, Boolean, Boolean. Age is numerical and sex and chronic disease binary are booleans.

In order to evaluate the model I will create a classifier measurements object and access a number of the available classifier measurements.

The classifier measurements I will use here are :

- Accuracy : The classification accuracy of the classifier. This is the fraction of classifications which are correct.
- Error : The error rate of the classifier. Accuracy + Error = 1.
- Precision : This is the proportion of positive classifications which are correct.
- Recall : This is the true positive rate.
- F1 Score : provides an average measure of recall and precision. A measure of models accuracy.

In[84]:=
```
measurementsNN = ClassifierMeasurements[nnFun,
   Values[testData, #Features &] -> Values[testData, #Objective &]];
nnRes = Labeled[TextGrid[{{"Accuracy", "Error", "Precision",
     "Recall", "F1Score"}, { measurementsNN["Accuracy"],
     measurementsNN["Error"], measurementsNN["Precision"],
     measurementsNN["Recall"], measurementsNN["F1Score"]}},
   Frame → All], "Neural Network Classifier Measures", Top]
```

Neural Network Classifier Measures

Out[85]=

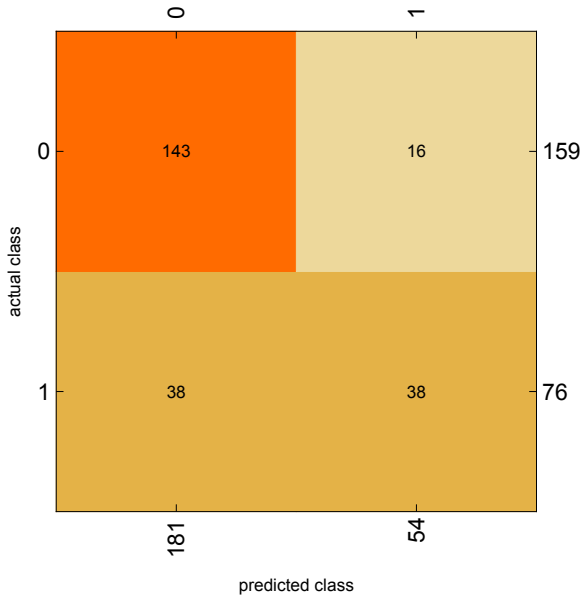
Accuracy	Error	Precision	Recall	F1Score
0.770213	0.229787	⟨| 0 → 0.790055, 1 → 0.703704 |⟩	⟨| 0 → 0.899371, 1 → 0.5 |⟩	⟨| 0 → 0.841176, 1 → 0.584615 |⟩

In[86]:=
```
measurementsNN["Report"]
```

Out[86]=

Classifier Measurements

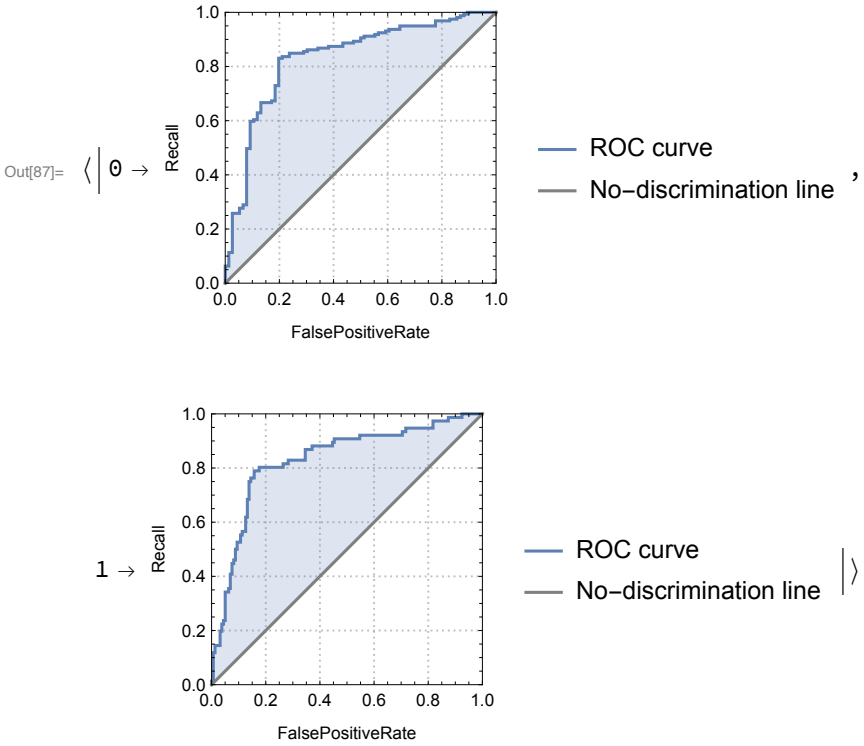
Number of test examples	235
Accuracy	(77.0 ± 2.8)%
Accuracy baseline	(67.7 ± 3.1)%
Geometric mean of probabilities	0.615 ± 0.027
Mean cross entropy	0.486 ± 0.043
Single evaluation time	4.27 ms/example
Batch evaluation speed	27.9 examples/ms
Rejection rate	0%



The confusion matrix shows that 179 of the 289 examples which were of class 1 were predicted as class 0. This might be due to the fact there there is a large imbalance in the data and not many examples of class 1.

In[87]:=
```
measurementsKNN["ROCCurve"]
```

Out[87]=



The ROC curve for both classes are strongly up to the top left which indicates good performance. The diagonal line at 45 degrees represent a randomised classification.

Model Comparison

All three models have been trained and used to classify the test set. Now I will compare the performance statistics of the three models to
determine which model performed best out of the three models.

Nearest Neighbour Results

In[88]:=
```
knnRes
```

Out[88]=

Nearest Neighbour Classifier Measures

Accuracy	Error	Precision	Recall	F1Score
0.778723	0.221277	<| 0 → 0.809249, 1 → 0.693548 |>	<| 0 → 0.880503, 1 → 0.565789 |>	<| 0 → 0.843373, 1 → 0.623188 |>

In[89]:=

Logistic Regression Results

Out[89]=
```
Logistic Regression Results
```

In[90]:=
```
lrRes
```

Out[90]=

Logistic Regression Measures

Accuracy	Error	Precision	Recall	F1Score
0.782979	0.217021	<| 0 → 0.803371, 1 → 0.719298 |>	<| 0 → 0.899371, 1 → 0.539474 |>	<| 0 → 0.848665, 1 → 0.616541 |>

Neural Network Results

In[91]:= **nnRes**

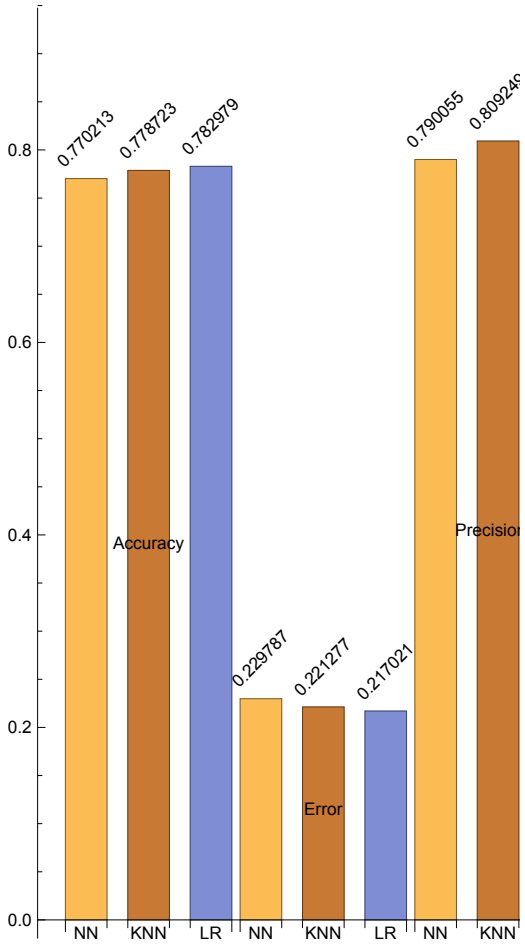
Out[91]=

Neural Network Classifier Measures

Accuracy	Error	Precision	Recall	F1Score
0.770213	0.229787	⟨| 0 → 0.790055, 1 → 0.703704 |⟩	⟨| 0 → 0.899371, 1 → 0.5 |⟩	⟨| 0 → 0.841176, 1 → 0.584615 |⟩

In[92]:=
```
BarChart[Transpose[{
    {measurementsNN["Accuracy"], measurementsNN["Error"],
     measurementsNN["Precision"][0], measurementsNN["Precision"][1],
     measurementsNN["Recall"][0], measurementsNN["Recall"][1],
     measurementsNN["F1Score"][0], measurementsNN["F1Score"][1]},
    {measurementsKNN["Accuracy"], measurementsKNN["Error"],
     measurementsKNN["Precision"][0], measurementsKNN["Precision"][1],
     measurementsKNN["Recall"][0], measurementsKNN["Recall"][1],
     measurementsKNN["F1Score"][0], measurementsKNN["F1Score"][1]},
    {measurementslr["Accuracy"], measurementslr["Error"],
     measurementslr["Precision"][0], measurementslr["Precision"][1],
     measurementslr["Recall"][0], measurementslr["Recall"][1],
     measurementslr["F1Score"][0], measurementslr["F1Score"][1]}}],
 LabelingFunction → (Placed[#, Above, Rotate[#, Pi / 4] &] &),
 ChartLabels →
  {Placed[{"Accuracy", "Error", "Precision 0", "Precision 1", "Recall 0",
     "Recall 1", "F1 Score 0", "F1 Score 1"}, Center], {"NN", "KNN", "LR"}},
 ChartLegends → {"Neural Network", "K Nearest Neighbours",
   "Logistic Regression"}, BarSpacing → Large, ImageSize → {1500, 500}]
```

Out[92]=



The accuracy was very similarly across all classifiers. The nearest neighbour classifier has the highest accuracy at 0.79371.

In Precision the neural network is highest for the class = 0 and knn is highest for the class = 1.
In Recall knn is highest for class = 0 and neural network is significantly highest for class = 1 .
For F1 score the knn classifier is slightly higher than the neural net. For class = 1 the neural network is significantly highest at 0.484988.
F1 score is a good overall measure here and would indicate that the neural network performed best under these evaluation metrics.

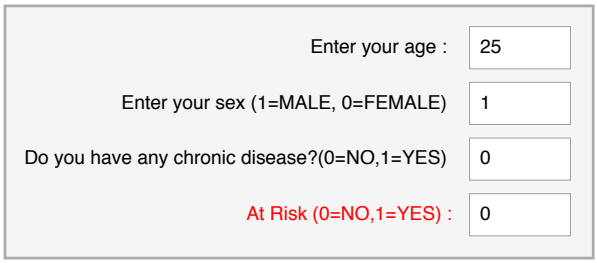
Visualisations

Calculator

People who are highly vulnerable to covid 19 and have underlying conditions are labeled as at risk. These people were at times advised to stay home and isolate.
A useful tool would be a risk calculator where people could input their information and be given a calculation of their level of vulnerability to the virus,
and they would know whether or not they should isolate. The model I have trained predicts binary outcomes but with better data including accurate outcome
values and consistent lists of symptoms and chronic diseases, a much more comprehensive calculator could be created similarly to this.

In[93]:=
```
Labeled[DynamicModule[{fun = nnFun},
  Deploy[Style[Panel[Grid[Transpose[{{Style["Enter your age : "],
         Style["Enter your sex (1=MALE, 0=FEMALE) "],
         Style["Do you have any chronic disease?(0=NO,1=YES) "], Style[
          "At Risk (0=NO,1=YES) :", Red]}, {InputField[Dynamic[a], Number],
         InputField[Dynamic[b], Number], InputField[Dynamic[c], Number],
         InputField[Dynamic[fun[{a, b, c}]], Enabled → False]}}],
      Alignment → Right], ImageMargins → 10], DefaultOptions →
    {InputField → {ContinuousAction → True,
       FieldSize → {{5, 30}, {1, Infinity}}}}]]], "Covid-19 At Risk Calculator"]
```

Out[93]=



Covid-19 At Risk Calculator

Cases Map

In the dataset there are latitude and longitude coordinates for each case. This is an interactive map of the cases in the dataset.
The dataset is 2.6 million records and when it is all used here the load time is extremely long. For this reason this map displays

1000 cases currently but can scale.

First pair the latitude and longitude values from the two columns :

In[94]:=
```
samp = RandomSample[testset, 10 000];
```

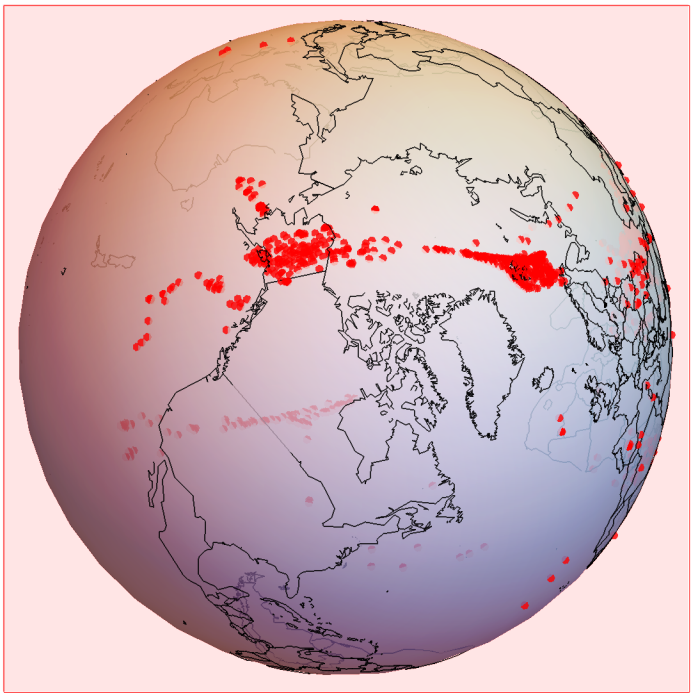
In[95]:=
```
l1 = Normal[samp[All, "longitude"]];
```

In[96]:=
```
l2 = Normal[samp[All, "latitude"]];
llpairs = Transpose[{l1, l2}];
```

In[98]:=
```
latlongF[{lat_, lon_}] :=
  r {Cos[lon °] Cos[lat °], Sin[lon °] Cos[lat °], Sin[lat °]};
```

In[99]:=
```
r = 6367.5; places = CountryData["Countries"];
Graphics3D[{Opacity[.8], Sphere[{0, 0, 0}, r],
  Map[Line[Map[latlongF, CountryData[#, "SchematicCoordinates"], {-2}]] &,
   places], {Red, PointSize[Medium], Point[latlongF[#]] & /@ llpairs}},
 Boxed → False, SphericalRegion → True, ViewAngle → 0.3]
```

Out[100]=



Travel History

Due to Covid - 19 travel has come to a halt. This is for obvious reason being that travel can enable the spread of the virus and also because exposure to large
numbers of people increases the chances of becoming infected. For this reason contact tracing has been explored for its potential uses in reducing the spread of the virus.

The column travel_history location contains a list of the places that the case had travelled to.

In[101]:=
```
hist = testset[Select[#"travel_history_location" ≠ "" &], All];
```

In[102]:=
```
hist1 = RandomSample[hist, 1000];
```

Each person's list of travel history locations can be seen as a path. These are all of the 772 different

paths that were taken:

In[103]:=
```
GroupBy[hist, "travel_history_location", Length]
```

Out[103]=

China	2
Harbin City, Heilongjiang	2
Thailand	3
Vietnam	4
Beijing, Beijing, China	1
Shenyang City, Liaoning; Tonghua City, Jilin	2
Meihekou City	1
Dunhua City	1
Qitaihe City, Heilongjiang	2
Wuhan	1
Gansu	1
Anhui	1
Aichi Prefecture	1
Guiyang City, Guizhou; Wuhan City, Hubei; Zhaotong City, Yunnan	1
Huangshi City, Hubei; Wuhan City, Hubei; Guiyang City, Guizhou	2
Kunming City, Yunnan; Hunan	1
Guangzhou City, Guangdong	8
Henan	4
Shilin Yi County, Kunming City, Yunnan; Chuxiong Prefecture, Yunnan; Dali Prefecture, Yunnan	2
Wuhan City, Hubei; Lijiang City, Yunnan	2

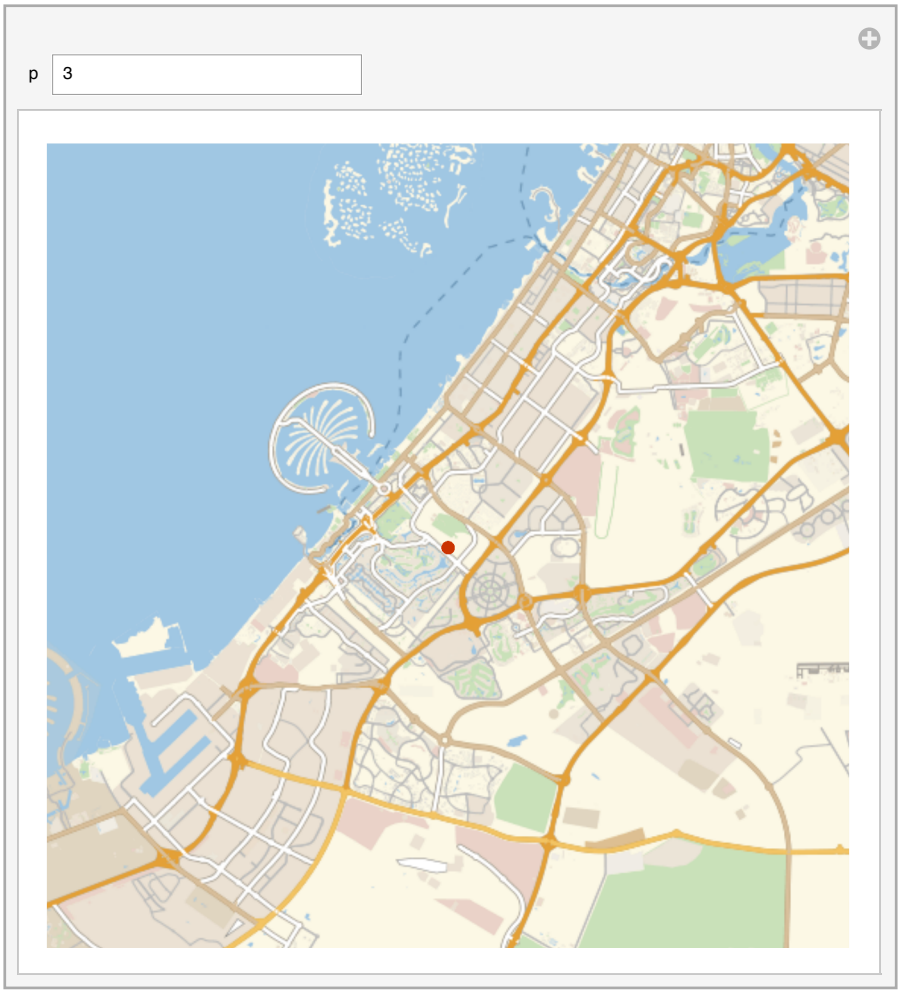
rows 1–20 of 510

By splitting each route on the “;” that separates each point on the route, I get a commma separated list of the points on the route:

By entering the ID of the person who’s travel history is needed this map updates to show the places they have traveled to.

In[104]:=
```
Manipulate[GeoListPlot[
  Map[ FindGeoLocation, StringSplit[hist1[p, "travel_history_location"], ";"]],
  Joined -> True], {p, 1}, ControlType → InputField]
```

Out[104]=



Conclusions

In this section I will briefly discuss the project in retrospect.

A factor that limited this project and limits any analyses of covid-19 patient level data is the lack of availability of the data. Covid-19
is a new virus, with a short data collection period. Privacy of individuals is also a concern in today's world and this also limits public availability
of patient level data. All data used in this project was anonymized and so there are no related privacy concerns. within this dataset there was a
lot of missing or inconsistent data which again limited the model training.

A significant improvement to the model's complexity would come from availability of patient's chronic diseases as well as symptoms. This would also
improve the effectiveness of the Covid-19 Risk calculator.

The Neural Network Classifier outperformed the Nearest Neighbour and the Logistic Regression Classifiers on this sample.

There are many useful visualisations and tools to be created from patient level data that can be used for a wide variety of tasks
as shown in the visualisations section of this project.