IST 707-HW4

spachika@syr.edu | suid:858281193

Submitted BY

**srinivas reddy pachika**

2020

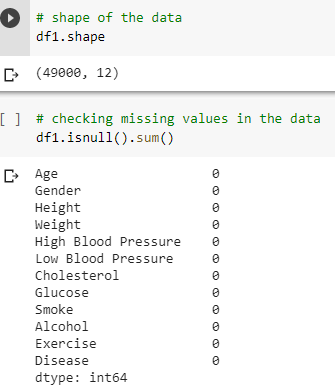
Om namah shivaya

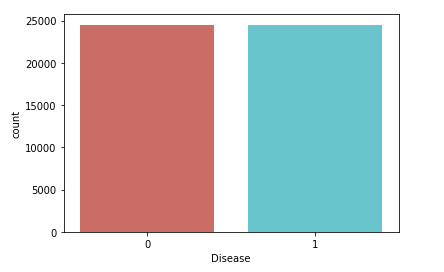
**SECTION 1: DATA PREPARATION**

In the Data preparation I have done four steps

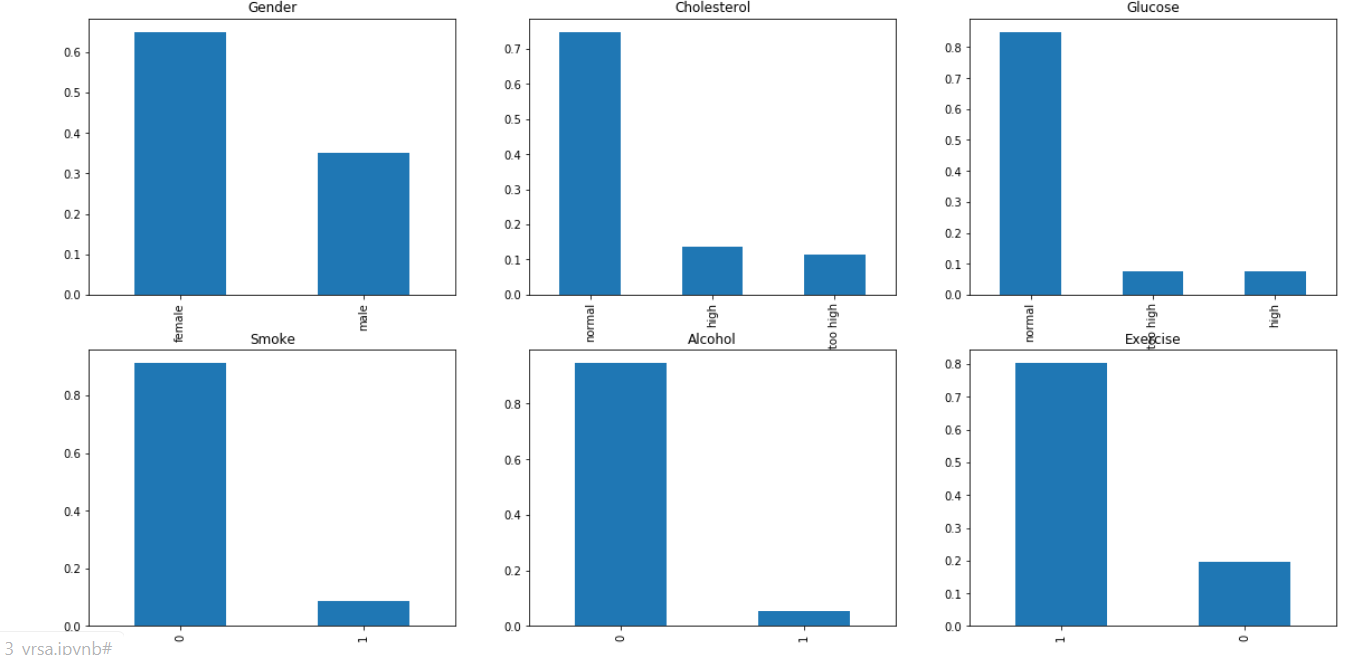
1. Handling missing values
2. Checking if the data set is balanced
3. Exploratory data analysis
4. Handling Outliers
5. Converting categories to numbers
6. Bring all the variables in range 0 to 1

* First step I have done after importing my data set is to see the shape od dataset and then check for the missing values, I found that the shape of my training data is (49000 rows, 12 columns ) and There are no missing values in the data set .



* As the next step I checked if the labels in my training data are balanced or not and found my data set is almost perfectly balanced with 50-50 distribution.  
   
* My exploratory data analysis and handling outliers is similar to my last assignment, l used frequency table or bar plots for categorical features which will calculate the number of each category in a variable. For numerical features, I used probability density plots to look at the distribution of the variables.

**Categorical Features:**



1. **It can be inferred from the above bar plots that:**
2. **65% patients in the dataset are male.**
3. **Around 75% of the patients in the dataset have normal cholesterol.**
4. **Around 85% patients in the dataset have normal glucose.**
5. **Around 10 % patients in the dataset smoke.**
6. **Around 5% patients in the dataset drink alcohol.**
7. **Around 80% patients in the dataset exercise.**

**Numerical Features:**

**Age:** we can observe from below plots that age variable is through not perfectly normal. Distribution looks decently normal with few outliers which are meaningful because there might me some patients who are very young and are below 30 years old.

**A picture containing window

Description automatically generated**

**A close up of a logo

Description automatically generated**

**Height and weight:**

It can be inferred from above plots that height and weight variables are also not perfect normal distributions but looks decent.

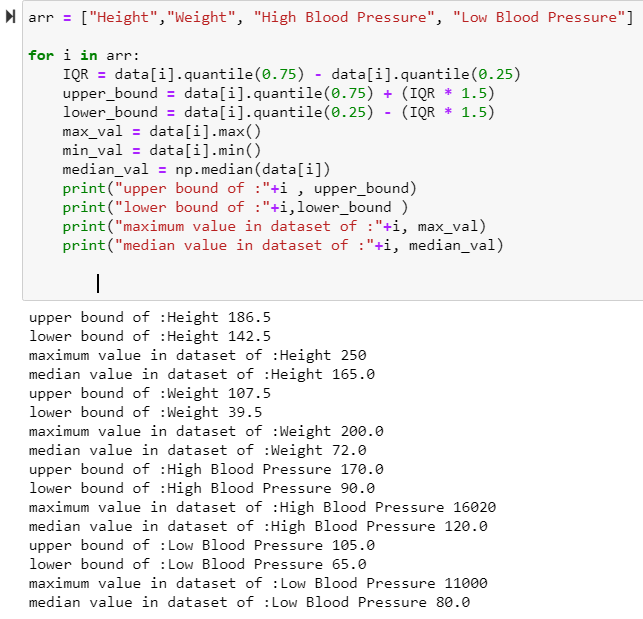
The above plot confirms the presence of outliers/extreme values for both height and weight. This can be attributed to the height and weight differences for different people in the society. Also, since the outliers in these both variables are meaningful, and they might not be an error. For example, some outliers in Hight are around 200 and it is possible that people can have height of 200cm similarly outliers’ values around 200 in weight can also be explained by the fact that people may weight around 200lbs.

**High Blood Pressure and Low Blood Pressure:**

The variables Hight blood pressure and low blood pressure have high skewness in the distribution and the outliers have no meaning in this case. A person cannot have values of low blood pressure and high blood pressure in thousands and hence these two variables have lot outliers which are errors and hence we have to use some techniques to treat outliers from both of them.

**Handling Outliers:**

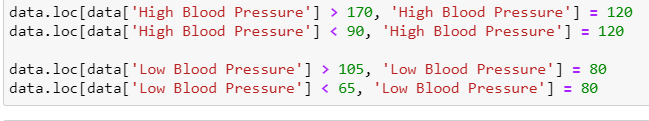
* I wanted to see the outliers of numerical variables in detail and hence I printed the upper bound and lower bound of the box plots of each of the numerical variables along with their maximum value and median.
* I tried to compare the maximum value of these variables with median value and see if the difference is high and unusual.



If we observe the above results the median value and maximum value of features height and weight , for height standard deviation is approx. 7, that is the max value is 8 standard deviations more , which is not too bad because there can be some very tall people in our population and these outliers may not be wrong observations. same is the case with feature weight, whose max value is 9 standard deviations more than median which also is possible because some patients may have overweight problem.

But when it comes to High blood pressure, max value is 16020 and median is 120, which is something wrong. a patient cannot have 16000 as his blood pressure value. from plots we can observe that many outliers are close to this value of 16000, which means all these values are errors similarly for low blood pressure the max value is 11000 whereas median is 80, and from plots observations many other outliers are close to this max value. These very high values are errors because humans cannot have such high values as low blood pressures.

**Therefore, for both High blood pressure and low pressure i will be treating my outliers by imputing any values more than the upper bound and less than the lower bound by median values of their respective columns**



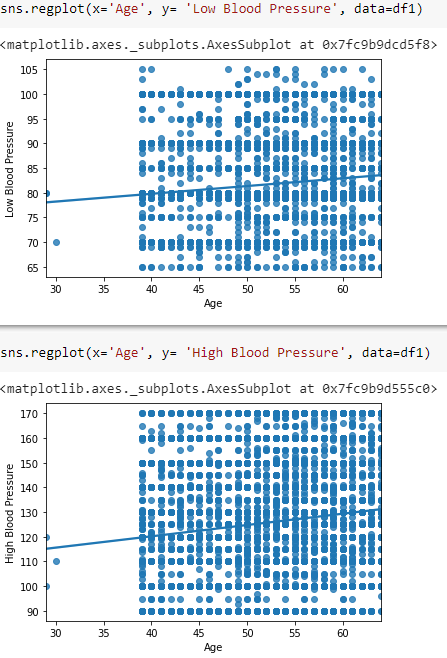
Then I plotted boxplots for these two variables to see if everything looks ok.

A close up of a logo

Description automatically generated

We see from above plot that there are no more outliers in these two variables.

* Then I tried to check if there is any linear relation between Age and High blood pressure and Age and Low blood pressure by fitting a regression line, Though I observed some positive correlation between these variables , the relation ship is not perfectly linear



* Then I checked correlation between the variables by plotting a correlation matric between the variables

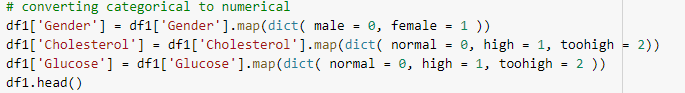
I observed that there is significant correlation between

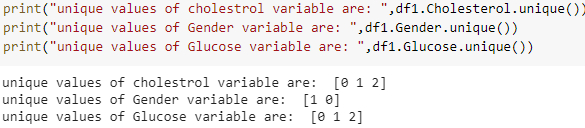
1. High blood pressure and Low blood pressure
2. High blood pressure and Disease
3. Low Blood Pressure and Disease
4. Age and Disease
5. BMI has correlation with height and weight
6. HBPLBP has correlation with High blood pressure and Low blood pressure

A picture containing different, small, display, group

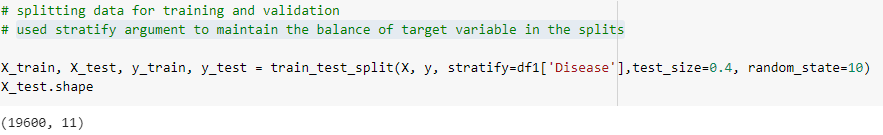
Description automatically generated

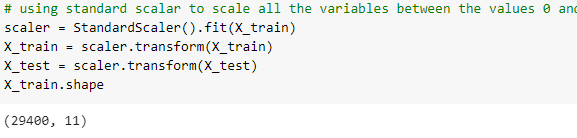
* Since most of machine learning algorithms with just one or two as exceptions could take only numerical features as inputs, as a next step I have converted my categorical features to numerical. For ordinal variables I have converted them to numerical features maintaining their order by mapping to the digits in the Ascending order.

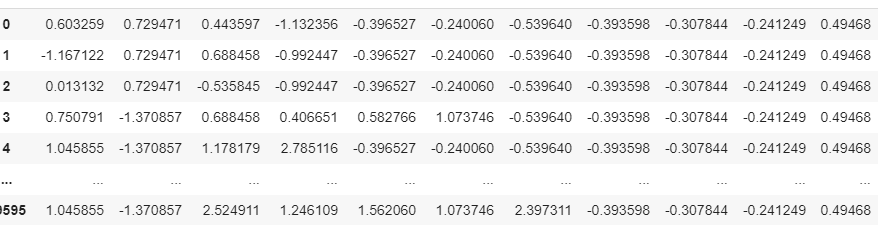




* As the next step in order to bring all the variables in range 0 to 1, I have used standardscalar () function after splitting my data into train and test with test split of 0.4. Though my data is almost perfectly balanced I used stratify argument to maintain the balance of target variable in the splits and account for any little imbalances.
* Standardization is very important in neural network because keeping the same scale for all our independent variables helps reduce the computational time and helps the neural network converge faster. Even in the case of logistic regression we can choose our optimizer like 'lbfgs','newton-cg','liblinear','sag','saga' and standardizing the features makes the convergence faster.







**CHOOSING EVALUATION METRIC:**

* I have chosen “recall” as my Evaluation metric because

1) I wanted to limit false negatives

2) I wanted to penalize FN more than FP

In this particular use case we are trying to predict a Disease of patient, For me here false negatives are far more important than false positives because if a patient who actually has disease(Actual positive ) if classified as Negative can even die if not treated , which is very bad, But if a patient who does not have disease (Actual Negative) if diagnosed or classified as positive, Though may face some side effects if given un necessary treatment but will not die or this situation is not as catastrophic as not treating an actual ill patient.

**LOGISTIC REGRESSION:**

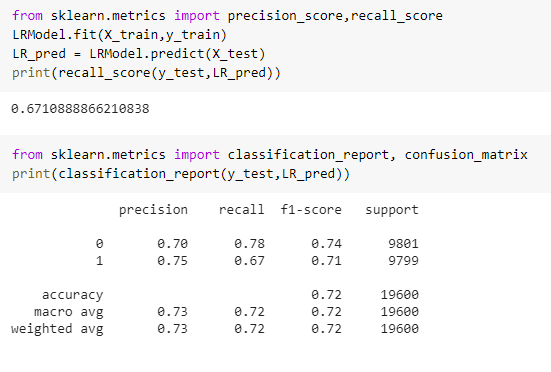
* Three important parameters in the logistic regression are C, Maxiter and Solver

**C** : This is Inverse of regularization strength and takes any positive float values , default=1.0

**max\_iter:** This is Maximum number of iterations taken for the solvers to converge, default=100

**Solver:** Algorithm to use in the optimization problem, Options : {‘newton-cg’, ‘lbfgs’, c ‘liblinear’, ‘sag’, ‘saga’}, Default = ‘lbfgs’.

**Logistic Regression Base Model.**



**My base model without any hyper parameter tuning gave me an accuracy of 72% and a recall score of 0.67.**

**Logistics Regression Hyper Parameter Tuning.**

* I have used grid search cv with following parameter options

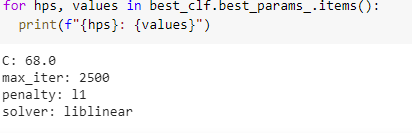
"C": from 1 to 100 with steps of 0.5

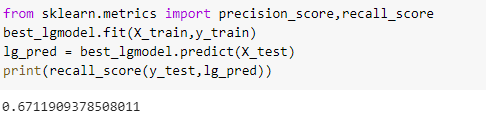
"penalty": l1 and l2

'solver' : 'lbfgs' and 'liblinear'

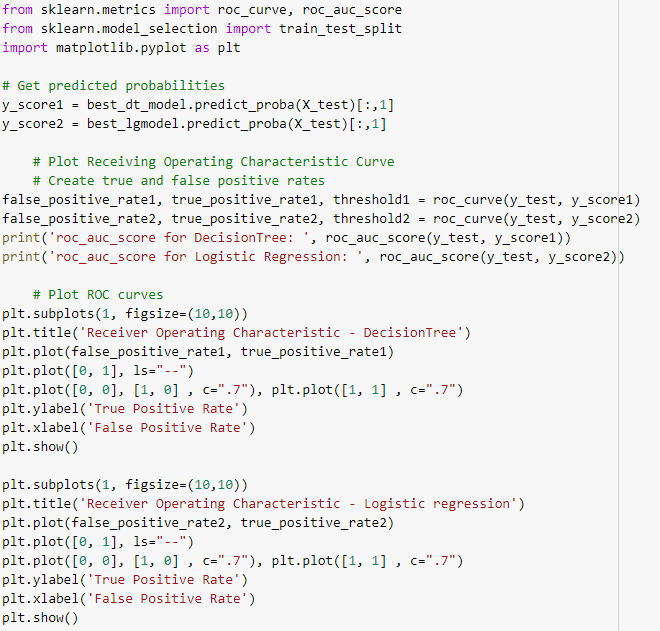
'max\_iter’: 100, 1000,2500, 5000

* Best Fit model parameters: after running grid search with 3 fold cross validation my best fit parameters are as shown below

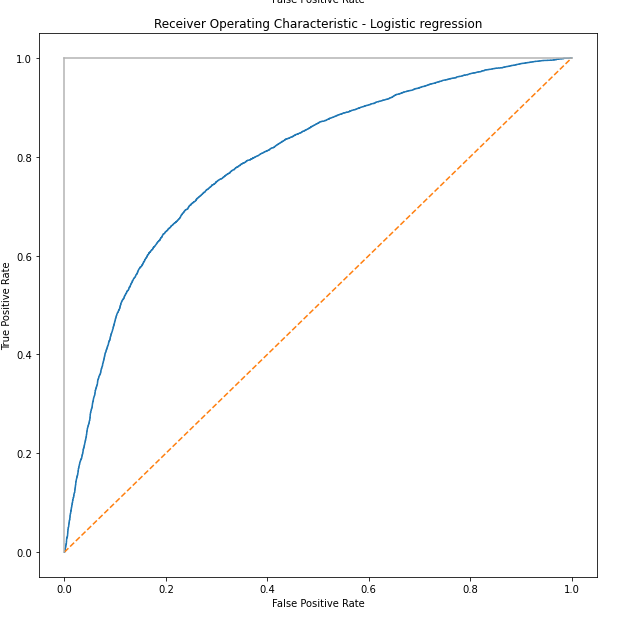


* My best model after hyper parameter tuning has got an accuracy of 0.6711  
    
  

**ROC & AUC**

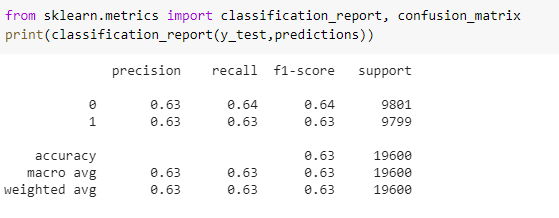






**Decision Tree – Base Model:**

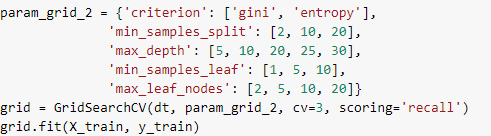
**My base model with out any tuning gave me results as below, My accuracy and recall scores were around 0.63**



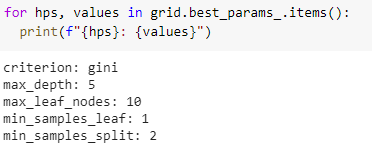
**Hyperparameter Tuning of decision tree**

**I have again used grid search cv with 3-fold cross validation for tuning my model.**

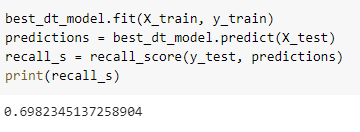
**Parameters that I choose for tuning as shown below:**



**My best fit model parameters:**



**Recall score of my best fit model is around 0.69**



**ROC & AUC**

**Code : I wrote combined code for plotting ROC curve for both logistic regression and decision tree and the code snippet is attached in the logistic regression section above.**



**A close up of a mans face

Description automatically generated**

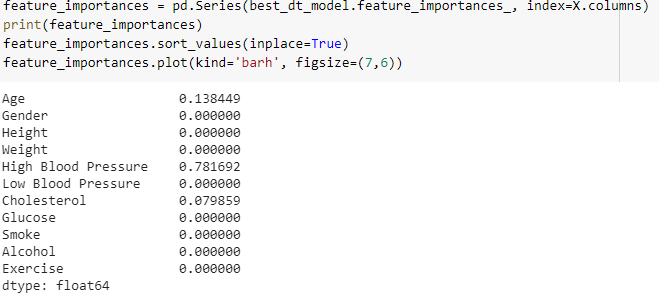
**Tree Plot for best model:**

**A close up of a sign

Description automatically generated**

**Feature Importance:**

**We can clearly see from our results that High blood pressure is the most important measure which will decide the probability of disease. This factor alone is almost 78% important in deciding whether a patient will have disease or not**

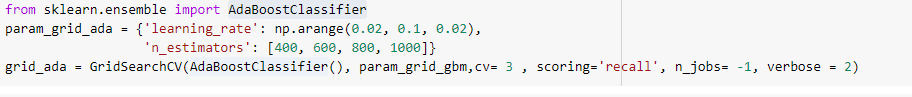


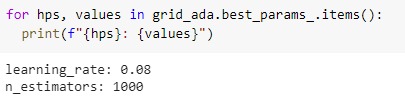
**A screenshot of a cell phone

Description automatically generated**

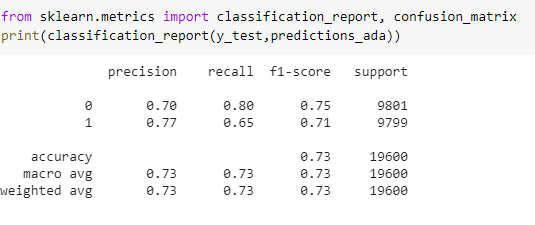
**AdaBoost:**

**I have used grid search cv to find best parameters for AdaBoost classifier with 3-fold cross validation. parameters for my best fit model are as below.**





**Recall score for my best fit model is around 0.65 and accuracy is around 0.73**



**ANN**

* The main parameter which I focused on while building my deep learning models are as follows.  
    
  **1)** **CHOOSING AN ACTIVATION FUNCTION**

For my hidden layers I choose my activation function as “relu” because of vanishing gradient of the sigmoid activation function. And for the final output layers as we are dealing with a binary classification model and my output is either zero or one, I choose sigmoid as activation function for my output layer.

In the back propagation the weights have to get updated to optimize our loss function. When we build models with multiple layers, during back propagation according to the chain rule we happen to take derivative of our activation function, and in case we choose sigmoid as our activation function in the hidden layers, the derivative of the sigmoid functions always lies between 0 and 0.25 and as we increase the number of layers the magnitude of the derivative of our loss with respect to old weights moves closer to zero. Due to which as we increase the number of layers our updated weights during back propagation will be approximately equal to our new weights, which is not desirable and hence to avoid this we choose relu over sigmoid as activation function in the hidden layers.

**2) DROPOUT OR REGULARIZATION**When we create model with multiple layers as we increase the number of nodes and layers, we run to high variance and overfitting problems and in order to avoid this we choose a dropout ration as one of our parameter which randomly deactivate few neurons in each epoch during forward propagation and avoid overfitting.  
  
I only chose dropout layer in my model with two hidden layers with a ration of 0.3 in both my first and second hidden layer

**3)Loss Function:**

I choose my loss function depending on the requirements just like we choose either RMSE or MSE for regression problems we choose binary cross entropy for binary classification problems and categorical cross entropy if we have more than two classes for classification.  
  
Since our problem has only two classes zero or 1, I choose binary cross entropy for as my loss function

**4) OPTIMIZATION**

After we come up with a cost function we choose an optimizer in order to reduce our cost function and converge to a global minima. Optimizers update the weight parameters to minimize the loss function. I have used mainly two optimizers in my models

**Stochastic Gradient Descent**

It is a variant of Gradient Descent. It tries to update the model’s parameters more frequently. In this, the model parameters are altered after computation of loss on each training example. So, if my dataset contains 500 rows SGD will update the model parameters 500 times in one cycle of dataset instead of one time as in Gradient Descent.

**Adam (Adaptive Moment Estimation)**

Adam optimizer works with momentums of first and second order. The intuition behind the Adam is that if we roll over the gradient so fast we may miss or jump over the minimum.

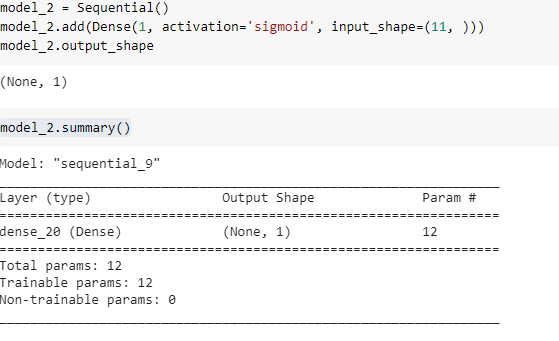
**5) Batch Size:**

The **batch size** defines the number of samples that will be propagated through the network.

For instance, let's say you have 1050 training samples and you want to set up a batch\_size equal to 100. The algorithm takes the first 100 samples (from 1st to 100th) from the training dataset and trains the network. Next, it takes the second 100 samples (from 101st to 200th) and trains the network again.

I chose my batch size as 500 in my models.

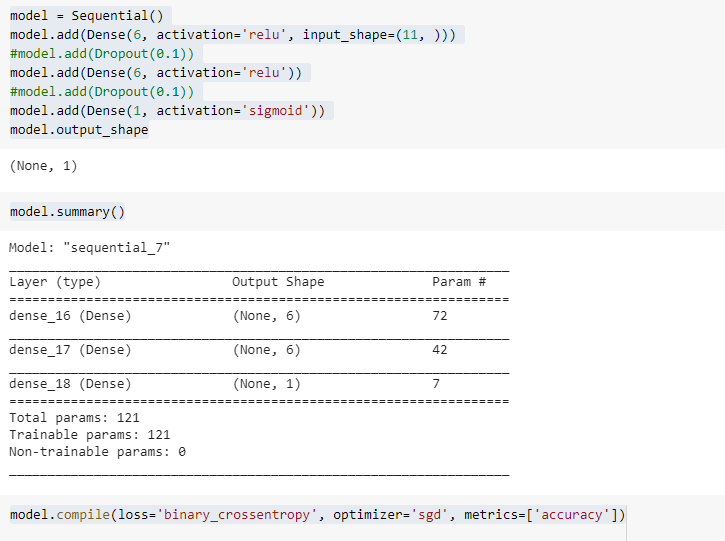
**Ann with ZERO hidden layers:**



**MY FINAL ACCURACY FOR THIS MODEL WAS AROUND 72%**



**ANN WITH TWO HIDDEN LAYERS**



**ACCURACY WITH SGD OPTIMIZER = 0.69 (WITH 20 EPOCHS )**



**ACCURACY WITH ADAM OPTIMIZER = 0.73**



A close up of a device

Description automatically generated

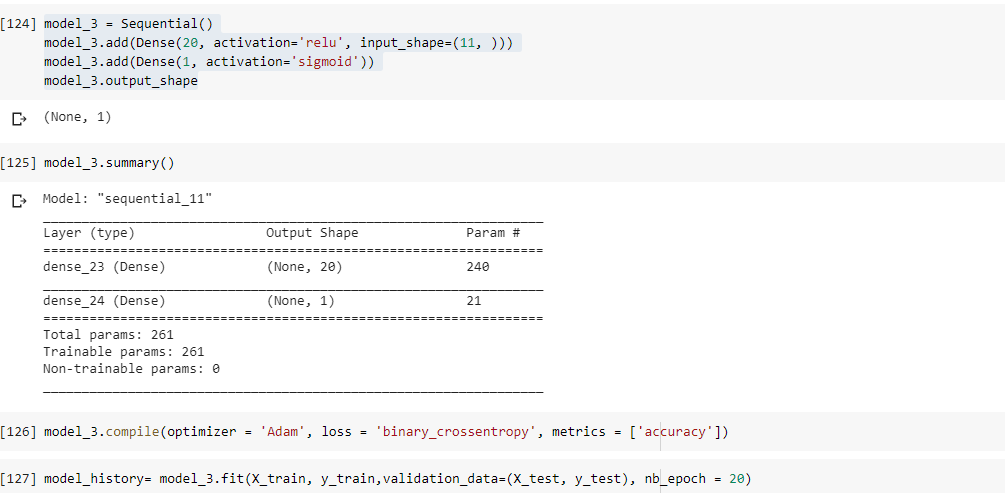
**Loss vs epoch graph for ANN2**

A close up of a map

Description automatically generated

**Accuracy vs epoch for ANN 2**

**ANN WITH ONE HIDDEN LAYER**



**ACURACY WITH ADAM OPTIMIZER FOR ANN1 = 0.73**

