Name: Akash Srivastava

Student ID: 17201082

For this report initial criteria for dataset is followed.

Problem statement:

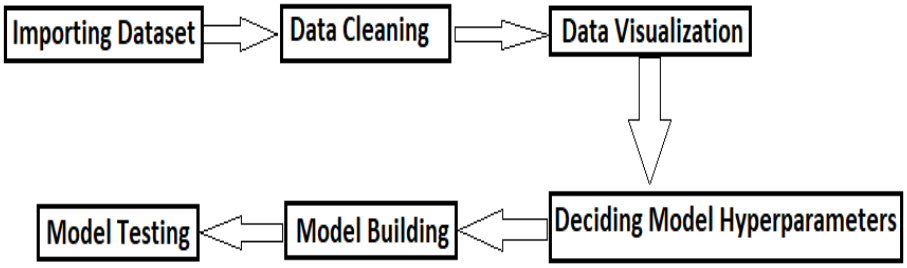
The problem I chose for the project is classification problem, which involves predicting whether a person is likely to have a chronic kidney disease or not based on various health parameters. There has been a constant rise in the number of people suffering from chronic kidney disease lately due to modern lifestyle. It is very difficult to detect kidney disease early and often involves surgery procedure as a cure. These procedure can cost a lot to the patients and this late detection of the disease may result in death of a patient, if proper treatment like dialysis or kidney transplant is not given to the patient. As there is rise in number of people suffering from kidney disease, there has been lot of research work going in the area. These research work are not only limited to health sciences but a lot of work is going on in the area machine learning to find patterns in various health parameters to detect whether a person is likely to have chronic disease in the future. These parameters can be age, sugar level, hypertension, anemia etc. The development of any such machine learning model may help in detection of chronic kidney disease at early stage which would in turn result in patients not losing huge amount for the treatment and may also help in saving lot of lives.

For the purpose of developing a machine learning model to predict whether a person is likely to have chronic disease in future the data is collected from the people with chronic kidney disease and from people without chronic kidney disease. This data includes various health parameters. Now on this data we apply deep learning to build the model that can effectively predict (based on the data) whether a person with some health parameter is likely to have chronic kidney disease in the future and if the model predicts predict yes, then suitable treatment can be started on that person.

For the purpose of building deep learning model, we will be using keras. Keras is chosen as it is a high level API to perform deep learning. Also keras is easy to understand and read as it uses tensor flow or theano at backend. Hence keras hides low level details of deep learning network. We just need to define model hyper parameters to perform deep learning. For the visualization purpose bokeh is used, it is a python library that helps in building interactive visualizations that can be represented on a web browser. Bokeh can be used to draw complex visualizations using simple commands.

Implementation:

The following image shows the overall flow diagram for the project.



In addition to the above stated steps in the image following steps can also be included to implement the solution in the production environment:

- Saving the model (optional)
- Making prediction using the saved model (to be implemented in future)

Importing Dataset:

As stated in the above the first step is importing the dataset from a csv file. For the purpose of importing the dataset we have used pandas. With the help of pandas read_csv() method we can import data in csv format into pandas dataframe. For using pandas in python we need to import pandas before using. After importing the dataset we can procced to the next step that is data cleaning.

The following image shows the data (after importing using pandas) that is will be used for model building:

```
In [48]: df.head()
Out[48]:
    age  blood pressure  Specific gravity  albumin  sugar red blood cell  \
0  48.0            80.0             1.020      1.0    0.0             NaN
1   7.0            50.0             1.020      4.0    0.0             NaN
2  62.0            80.0             1.010      2.0    3.0          normal
3  48.0            70.0             1.005      4.0    0.0          normal
4  51.0            80.0             1.010      2.0    0.0          normal

   pus cell pus cell clumps     bacteria  blood glucose random       ...        \
0    normal       notpresent  notpresent                 121.0       ...
1    normal       notpresent  notpresent                   NaN       ...
2    normal       notpresent  notpresent                 423.0       ...
3  abnormal          present  notpresent                 117.0       ...
4    normal       notpresent  notpresent                 106.0       ...

   packed cell volume  white blood cell count  red blood cell count  \
0                44.0                  7800.0                   5.2
1                38.0                  6000.0                   NaN
2                31.0                  7500.0                   NaN
3                32.0                  6700.0                   3.9
4                35.0                  7300.0                   4.6

   hypertension  diabetes mellitus  coronary artery disease  appetite  \
0           yes                yes                       no      good
1            no                 no                       no      good
2            no                yes                       no      poor
3           yes                 no                       no      poor
4            no                 no                       no      good

   pedal edema anemia Target class
```

Data cleaning:

The next step after importing the data is data cleaning. Data cleaning is a very important step in building a machine learning model. The dataset we import may contain data that may prevent the model to make good prediction. The dataset may contain null values, outliers, redundant rows etc. Replacing the null values with some values, removing outliers and redundant rows are part of data cleaning. If these operations are not performed the resultant model will not predict the output efficiently. In addition to these data may have some categorical attributes and for a machine learning algorithm to understand them

we need to change these to some numerical values. Converting categorical attributes to numerical is also a part of data cleaning step.

The following image show the count of null values present in each attribute:

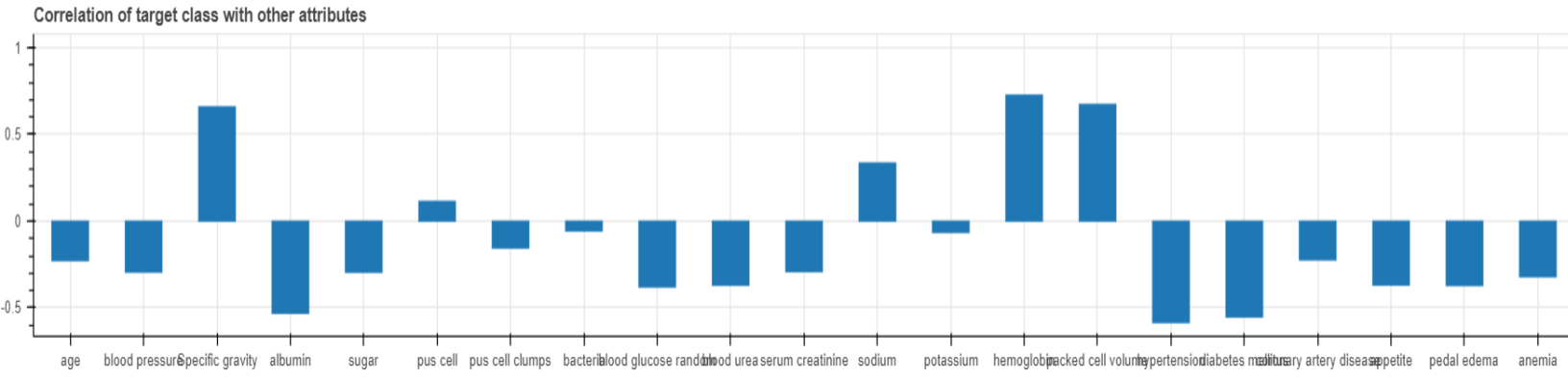
```
In [49]: df.isnull().sum()
Out[49]:
age                              9
blood pressure                  12
Specific gravity                47
albumin                         46
sugar                           49
red blood cell                 152
pus cell                        65
pus cell clumps                  4
bacteria                         4
blood glucose random            44
blood urea                      19
serum creatinine                17
sodium                          87
potassium                       88
hemoglobin                      52
packed cell volume              71
white blood cell count         106
red blood cell count           131
hypertension                     2
diabetes mellitus                2
coronary artery disease          2
appetite                         1
pedal edema                      1
anemia                           1
Target class                     0
dtype: int64
```

From the above image it is evident that the attributes like red blood cell, white blood cell count and red blood cell count have lot of null values. Therefore it is better to remove these attributes as replacing null values for these attributes. For the rest of the attributes the null values are replaced with some value. In case of numerical attribute median value for that attribute is used and for categorical attribute “unknown” is used.

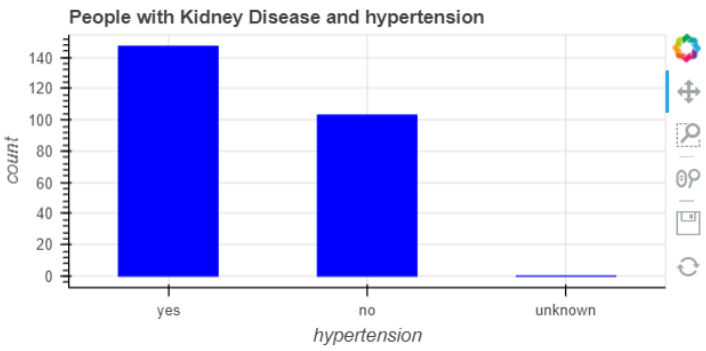
After handling the null values we changed the categorical attributes to numerical using LabelEncoder() from the sklearn.preprocessing package. After encoding the categorical attributes we then find correlation of the target class attribute with other attribute and remove those attributes that have weak correlation (In this case < |0.3|) with the target class attribute. Now we can use this cleaned data to build a machine learning model.

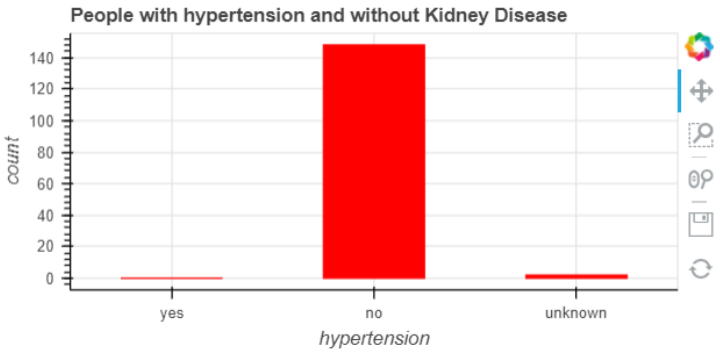
Data Visualization:

In order to decide the model hyper parameters, we first need to understand the data and for understanding the dataset usually visualizations are used. For the purpose of visualization, I used bokeh. The following are the visualizations that show the relation of the target call with other attributes:

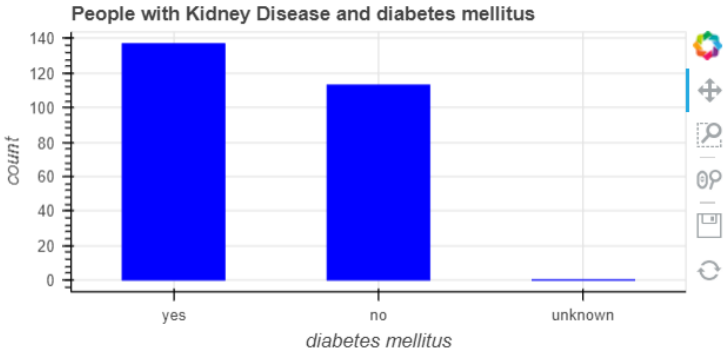


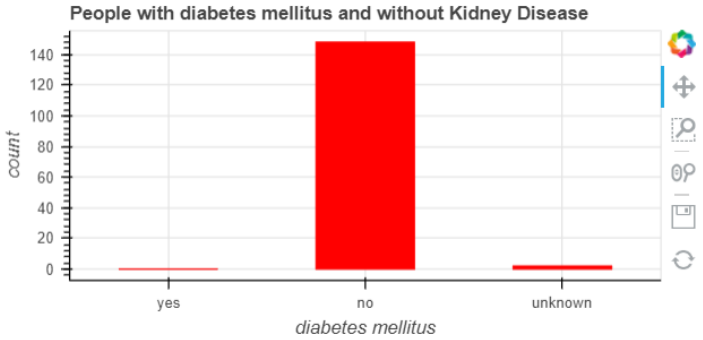
The above image shows the correlation of target class with other attributes.



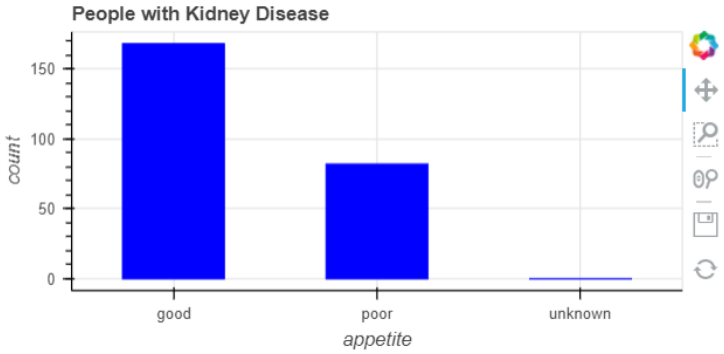


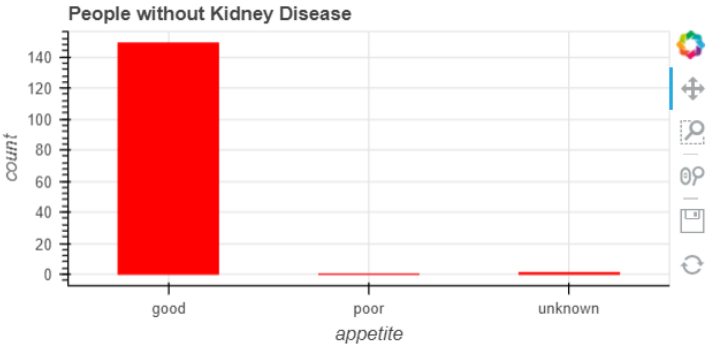
The above two visualization show the number of people with/without kidney disease and hypertension. From the above two visualizations it is clear that people with kidney disease may/may not have hypertension but people without kidney disease don't have hyper tension. The other thing to notice is that number of people with kidney disease and hypertension are more than number of people with kidney disease and without hypertension.





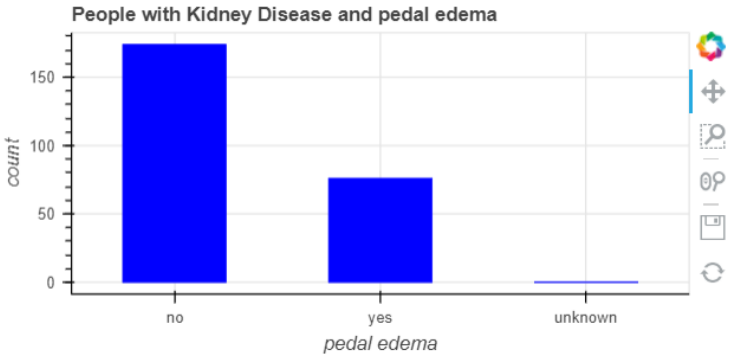
The above two visualization show the number of people with/without kidney disease and diabetes mellitus. From the above two visualizations it is evident that people with kidney disease may/ may not have diabetes mellitus but people without kidney disease don't have diabetes.

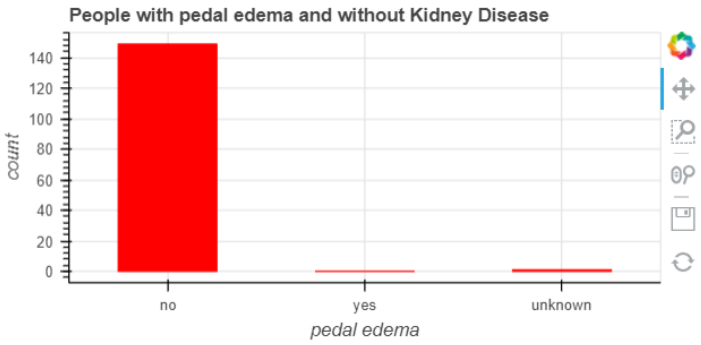




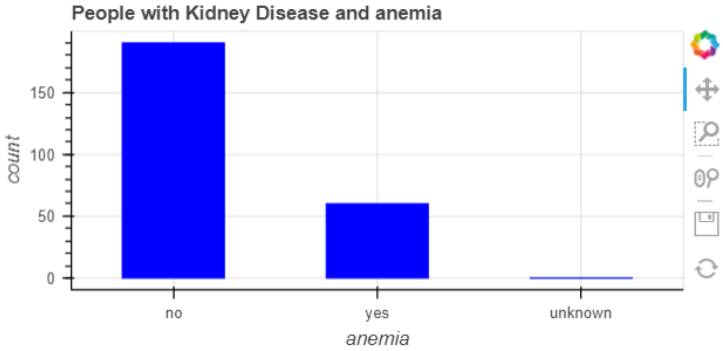
The above two visualization show the number of people with/without kidney disease and Good/Poor appetite. From the above two visualizations it is evident that people with kidney disease may/May not have good appetite but people without kidney disease have good appetite. The odd thing here is that the

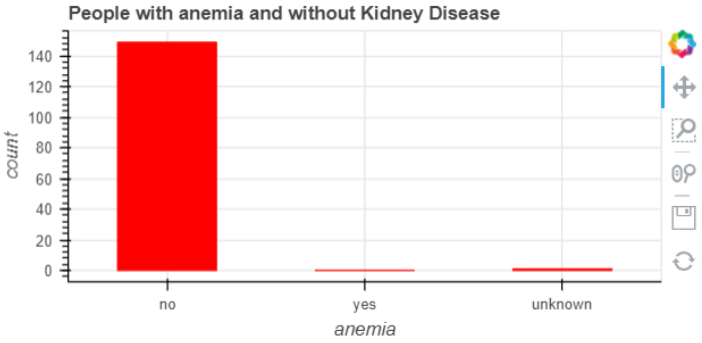
number of people with kidney disease and good appetite is more than the number of people with kidney disease and bad appetite.





From the above visualization it can be inferred that people with kidney disease and without pedal edema is almost equal to number of people without kidney disease and pedal edema. The other thing to notice here is that there is no person without kidney disease who has pedal edema.





From the above two visualization, the first one shows number of people with kidney disease and with/ without anemia and the second one shows number of people without kidney disease and with/without anemia. The thing to notice here is that no person without kidney disease is anemic.

Deciding Model Hyper parameter:

For a deep learning model hyper parameters are the configuration of a network. It includes number of layers present in the network, nodes present in each layer, activation function of each layer, loss function, optimizer function etc. These parameters decide the working of a deep learning network. In our case we are using keras to build our deep learning network. To use keras we first need to import it. Keras run on the top of tensor flow or Theano. I used sequential model to build the network and used one hidden layer with the input size of 13. The first layer has input size equal to the number of attributes in the training set because it takes input without any processing. This layer can have any number of nodes but it is advised to have number of nodes between any number between input size and output size. In our case, I chose number of nodes to be equal to the number of attributes present in the dataset. The activation function used in the

first layer is relu. I chose relu because it require less computation as it finds maximum value between two values. Therefore not much computation also better shows convergence**.**

The last layer in a deep learning network is the output layer and usually this layer has only one node which is the result of the deep learning network. In our case also this layer has one node and the activation function I chose for this layer is sigmoid. I have used sigmoid function as an activation function because it works on the probability and is used mostly in binary classification which is the case for us.

After designing the deep learning network, now is the time to decide the loss function and the optimizer that will be used for the model training the backward propagation phase. The loss function I used is binary cross entropy as it is best for binary classification problem. In the forward propagation phase the deep learning network makes the prediction and then calculates the error on this prediction based on the actual output and in the backward propagation it tries to make changes in the weights to reduce this error using some optimizer function. In our case, I am using Adam as an optimizer with the learning rate of 0.01. Adam is gradient descent algorithm that tries to find the minimum error by changing the weights at each layer using the learning rate defined.

Model Building:

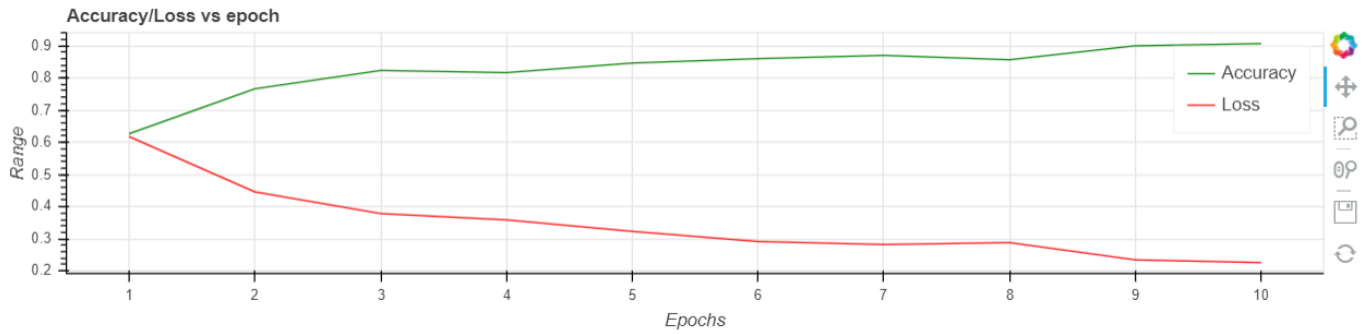
Model Building step of a deep learning problem involves implementing the hyper parameter and training the data using the training dataset. This step is one of the most important step in deep learning. In this step we decide for how many times the training on the training set will happen after updating the weights. In each epoch the training set is used to make prediction and loss is calculated using the prediction and the actual value. Now using this loss weights are updated. Deciding appropriate number of epochs is very important as if the number epochs in very low then the resulting model is said to be under fit and does not make good prediction on the other hand if the number of epochs are too large then the model over fits that is model memorizes the training set and while testing decreases. In our problem, after trying different values I chose 10 as number epochs. After deciding number of epochs the next step is to apply the model on the dataset. After building the model the next step is to assess how good the model is in predicting the values for an unknown new data and this step is called model testing.

Model Testing:

In this step we test the model build on the training data using the test data. This test data is usually part of the original dataset but is held out before training is done. This test data is new to the model and based on the predictions made by the model on this testing data the accuracy of the model is calculated. While testing the model usually the accuracy and the loss is considered. Following is the graph of the accuracy and the loss in each epoch of the model training:

From the above graph it can be seen as the number epoch increases the accuracy also increases and loss decreases.

The accuracy and loss for the testing data is as follow:




```
loss for testing set :  0.13826279401779173
Accuracy for testing set :  95.0 %
```

In addition to the above steps we can also save the model and use it from the application to predict how likely a person will have chronic kidney disease. This can be done using pickle package available in python.

Conclusion and Recommendations:

The above build model can be used to make prediction on whether a person is prone to have chronic kidney disease in future. This prediction can help in saving lot of lives lost due to kidney failure by giving proper treatment to the people who are likely to have chronic kidney disease. In addition to this it can be inferred from the visualizations above that people with hypertension, pedal edema, diabetes and anemia are more likely to have chronic kidney disease in the future. No person without chronic kidney disease has any of these condition. Considering the above health parameter regular checkups and preventive measures can be advised to the patients who fall in this category.