Jeet Shah

G41262537

Project 1 Report

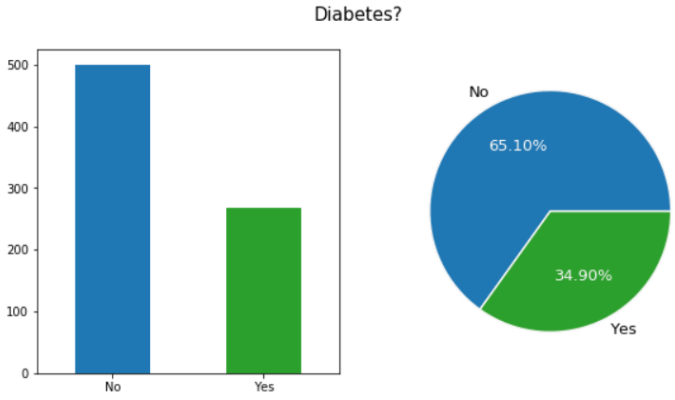
Dataset 1 (PIMA Indian Diabetes dataset)

Dataset Details:

- This dataset is obtained from the National Institute of Diabetes and Digestive and Kidney Diseases.
- The objective of the dataset is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. The datasets consists of several medical predictor variables and one target variable, Outcome.
- Features of the dataset are Insulin level, Age, Glucose level and many more. The outcome states whether the person has diabetes or not.

I have plotted some graphs that can give somewhat idea on the input data.

Graph 1:

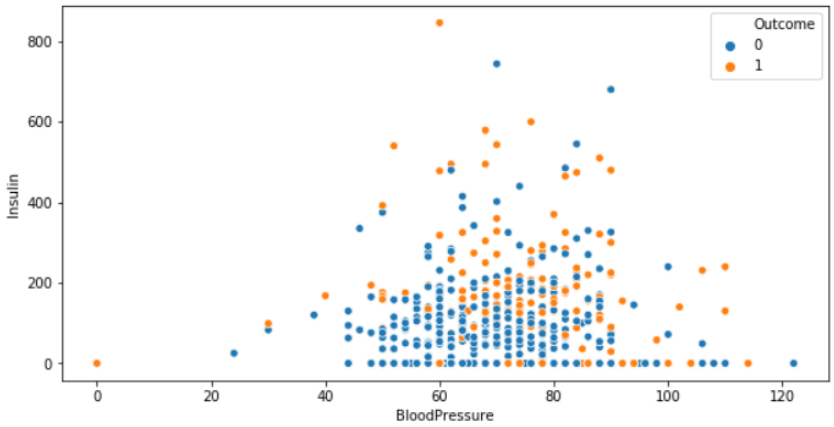


This graph shows that 65.10% people of the given dataset doesn't have diabetes.

Graph 2:

```
[9]: plt.figure(figsize=(10,5))
     sns.scatterplot(data=df,x="BloodPressure", y="Insulin",hue="Outcome")
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1f7ddbee8c8>
```



Here, we can see the blood pressure and Insulin level for both types of the people.

Algorithm description:

I have divided the input data in two parts. 85% of the given dataset is to train the model, as model needs more data to get trained, and the other 15% is to test it.

For the Pima Dataset, I have used 3 distance metrics. One is Euclidean, Second is Manhattan and the third is Hamming distance metric. The result of these 3 metrics for 5 nearest neighbor is shown below.

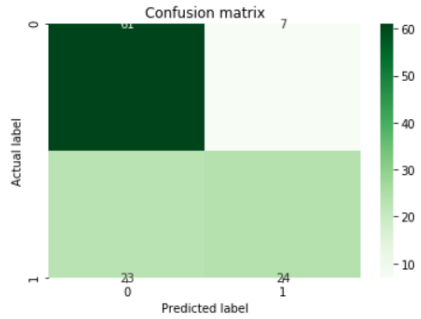
Algorithm Results and Runtime:

It takes more time for Euclidean algorithm. By increasing value of k (number of nearest neighbor) the accuracy decreases.

```
Enter the k(number of nearest neighbor) value: 5
Enter the type of matrix - Euclidean / Manhattan / Hamming: Hamming
Accuracy: 62.60869565217392
Time taken is: 0.23441553115844727

Enter the k(number of nearest neighbor) value: 5
Enter the type of matrix - Euclidean / Manhattan / Hamming: Manhattan
Accuracy: 37.391304347826086
Time taken is: 0.20046377182006836

Enter the k(number of nearest neighbor) value: 5
Enter the type of matrix - Euclidean / Manhattan / Hamming: Euclidean
Accuracy: 71.30434782608695
Time taken is: 0.35205817222595215
```

With Euclidean algorithm it takes more time. By running the code, we can see time does not vary much by changing value of k.

```
Confusion Matrix
[[61  7]
 [23 24]]
TP: 61 FN: 7
FP: 23 TN: 24

Text(0.5, 15.0, 'Predicted label')
```



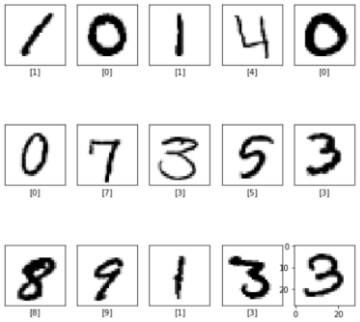
Dataset 2 (Digit Recognizer dataset)

Dataset Details:

In the training set, there are 785 colums. First column represents label and rest every column has a name pixelx, where x is an integer between 0 and 783, inclusive. We are given image data which is of 28*28 (784) pixels. Pixels value are from 0 to 255.

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel78
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	

One example of the numbers is shown below.

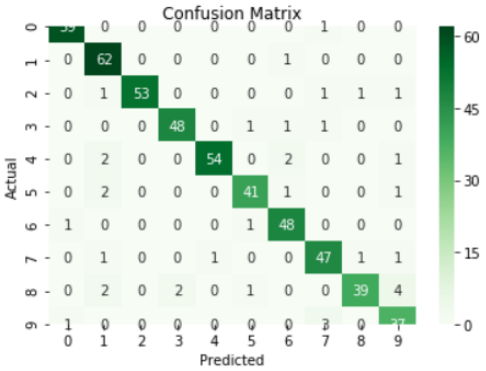


Algorithm description:

We have 42000 rows of training data. I have used 85% of the data as training data as more data is needed to train the model. To normalize input, I have divided pixel values by 255 as maximum value of the pixel is 255 and so pixels' range of value will be from 0 to 1. I have used Euclidean algorithm to calculate distance.

Algorithm results and run time:

```
Model accuracy: 92.95238095238095
Time taken: 13.651780366897583

Text(0.5, 15.0, 'Predicted')
```




```
Accuracy of number '1' for k = 3 is: 98.78048780487805
Time taken: 11.585018396377563
Accuracy of number '1' for k = 13 is: 97.58551307847083
Time taken: 12.695051193237305
Accuracy of number '1' for k = 23 is: 96.35627530364373
Time taken: 13.543782472610474
Accuracy of number '1' for k = 33 is: 95.67901234567901
Time taken: 13.172771215438843
Accuracy of number '1' for k = 43 is: 95.25773195876288
Time taken: 13.831608533859253
Accuracy of number '1' for k = 53 is: 94.375
Time taken: 13.102929830551147
Accuracy of number '1' for k = 63 is: 94.33962264150944
Time taken: 12.990261316299438
Accuracy of number '1' for k = 73 is: 93.94572025052193
Time taken: 11.842332124710083
```

We can see for larger value of k (number of nearest neighbor) the algorithm becomes a bit less accurate and also takes more time.