

## **AIML MINI PROJECT**

***GROUP MEMBERS:***

***SMRITHI AGRAWAL: 2019140001***

***KAREN CASTELINO: 2019140010***

***BATCH: A***

***CLASS; TE IT***

***COURSE: AIML LAB***

**AIM:** To detect various diseases in a person using various machine learning models.

**GITHUB LINK:** <https://github.com/smrithi1912/AIML>

**PROBLEM STATEMENT:**

In today's time we see a lot of the shortage of doctors in the world especially in India. A lot of people are suffering without a proper medical checkup. Also, most of the cases that arise lead to death due to lack of timely medical attention. So, to cope up with all of those problems, we have designed this web application that predicts if a person suffers from a particular disease or not using Machine Learning

**APPLICATIONS:**

- To remove the dependencies on the doctors
- To help out the poor and helpless people with the normal medical checkup
- To help people avoid paying huge amount to the doctors unnecessarily
- To extend the role of the technology in the medical field

**ALGORITHMS USED:**

1. Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression[1] (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

Model :

Output = 0 or 1

Hypothesis =>  $Z = WX + B$

$$h\Theta(x) = \text{sigmoid} ($$

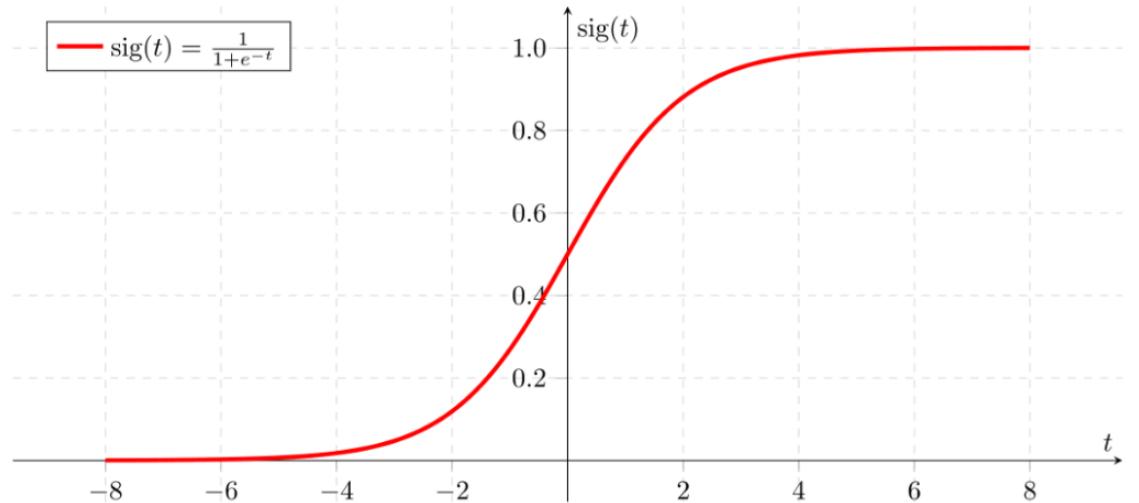
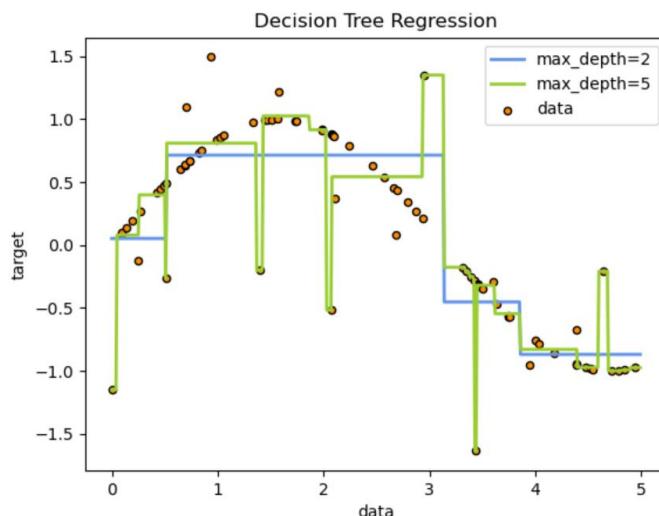


Figure 2: Sigmoid Activation Function

Analysis of the hypothesis: The output from the hypothesis is the estimated probability. This is used to infer how confident can predicted value be actual value when given an input X. Consider the below example,

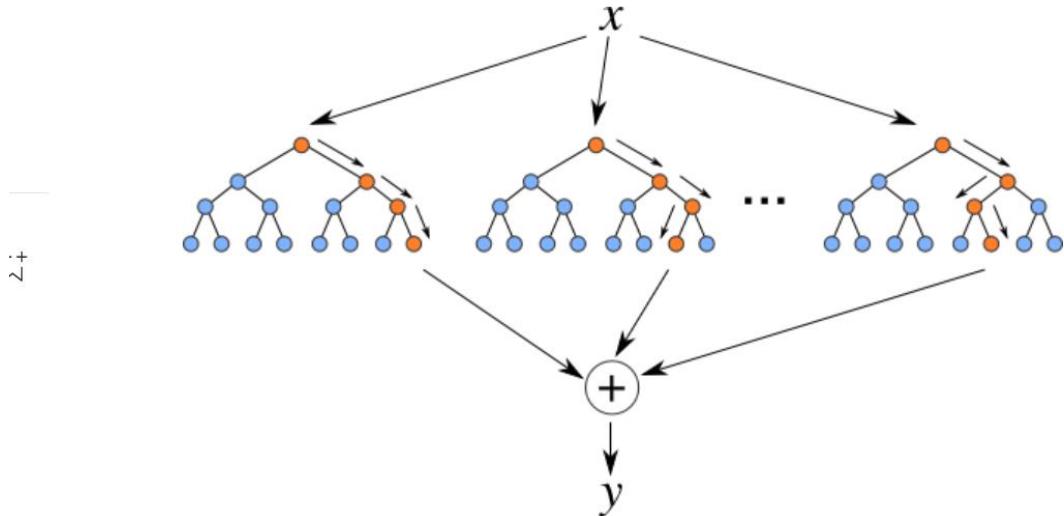
## 2. Decision Tree Regressor

Decision trees build regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.



### 3. Random Forest Regression

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. To get a better understanding of the Random Forest algorithm, let's walk through the steps:

- Pick at random  $k$  data points from the training set.
- Build a decision tree associated with these  $k$  data points.
- Choose the number  $N$  of trees you want to build and repeat steps 1 and 2.
- For a new data point, make each one of your  $N$ -tree trees predict the value of  $y$  for the data point in question and assign the new data point to the average across all of the predicted  $y$  values.

#### MODEL FILES USED:

Cancer Model = model  
Diabetes Model = model1  
Heart Model = model2

#### DATASETS USED:

##### Drive Link

Cancer = cancer.csv  
Diabetes = diabetes.csv  
Heart = heart.csv

#### TOOLS NEEDED:

Python  
Flask  
TensorFlow  
pillow

scikit  
pandas  
numpy  
HTML  
CSS

## Colab File:

The screenshot shows a Google Colab notebook titled "DiseasePrediction.ipynb". The code cell [1] imports numpy, pandas, matplotlib, and seaborn, and sets %matplotlib inline. It also prints the current directory and ignores warnings. The code cell [2] reads three CSV files into datasets: cancer.csv, diabetes.csv, and heart.csv.

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')

['.config', 'diabetes.csv', 'cancer.csv', 'heart.csv', 'sample_data']

[2]: dataset1 = pd.read_csv("cancer.csv")
dataset2 = pd.read_csv("diabetes.csv")
dataset3 = pd.read_csv("heart.csv")
```

The code cell [3] shows the type of dataset1, which is a pandas.core.frame.DataFrame. The code cell [5] shows the shape of dataset1, which is (569, 33). The code cell [6] shows the head of dataset1, displaying the first few rows of the dataset.

```
[3]: type(dataset1)
pandas.core.frame.DataFrame

[5]: dataset1.shape
(569, 33)

[6]: dataset1.head()
```

ractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave_points_worst	symmetry_worst	fractal_dimension_worst	Unamed: 32
0.006193	25.38	17.33	104.60	2019.0	0.1622	0.6556	0.7119	0.2654	0.4601	0.11690	NaN
0.003532	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	NaN

DiseasePrediction.ipynb - Colab

```
[12] dataset1=dataset1.drop("Unnamed: 32", 1)
[13] dataset1.sample(5)

[14] dataset1.describe()

[15] dataset1.info()
```

	radius_se	fractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave_points_worst	symmetry_worst	fractal_dimension_worst
0.01344	0.002669	17.32	17.76	109.60	928.2	0.1354	0.1361	0.19470	0.13570	0.2300	0.07230	
0.03176	0.002365	23.24	27.84	158.30	1656.0	0.1178	0.2920	0.38610	0.19200	0.2909	0.05965	
0.02104	0.001887	14.48	21.82	97.17	643.8	0.1312	0.2548	0.20900	0.10120	0.3549	0.08118	
0.01148	0.002379	17.18	18.22	112.00	905.6	0.1065	0.2791	0.31510	0.11470	0.2668	0.05273	
0.02572	0.002278	10.65	22.88	67.68	347.3	0.1265	0.1200	0.01005	0.02232	0.2262	0.06742	

	radius_se	fractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave_points_worst	symmetry_worst	fractal_dimension_worst
0.005266	0.002646	4.853242	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.065732	0.061867	0.018061	
0.007862	0.000995	7.930000	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.000000	0.156500	0.055040	
0.015160	0.002248	13.010000	21.080000	84.110000	515.300000	0.116600	0.147200	0.114500	0.064900	0.250400	0.071460	
0.019730	0.003187	14.970000	25.410000	97.660000	656.500000	0.131300	0.211900	0.226700	0.099930	0.262200	0.060040	
0.023460	0.004550	18.790000	29.720000	125.400000	1084.000000	0.146000	0.339100	0.362900	0.161400	0.317900	0.092000	
0.078950	0.029840	36.040000	45.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.291000	0.663800	0.207500	

```

DiseasePrediction.ipynb - Colab: + colab.research.google.com/drive/1oRvC2WjcvOtgX41WHh5Bn2GTHQNaUc#scrollTo=VD/RgjzfY0sa
File Edit View Insert Runtime Tools Help All changes saved
Files + Code + Text
dataset1.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               569 non-null    int64  
 1   diagnosis        569 non-null    object  
 2   radius_mean      569 non-null    float64 
 3   texture_mean     569 non-null    float64 
 4   perimeter_mean   569 non-null    float64 
 5   area_mean        569 non-null    float64 
 6   smoothness_mean  569 non-null    float64 
 7   compactness_mean 569 non-null    float64 
 8   concavity_mean   569 non-null    float64 
 9   concave_points_mean 569 non-null    float64 
 10  symmetry_mean   569 non-null    float64 
 11  fractal_dimension_mean 569 non-null    float64 
 12  radius_se        569 non-null    float64 
 13  texture_se       569 non-null    float64 
 14  perimeter_se     569 non-null    float64 
 15  area_se          569 non-null    float64 
 16  smoothness_se    569 non-null    float64 
 17  compactness_se   569 non-null    float64 
 18  concavity_se     569 non-null    float64 
 19  concave_points_se 569 non-null    float64 
 20  symmetry_se     569 non-null    float64 
 21  fractal_dimension_se 569 non-null    float64 
 22  radius_worst     569 non-null    float64 
 23  texture_worst    569 non-null    float64 
 24  perimeter_worst  569 non-null    float64 
 25  area_worst        569 non-null    float64 
 26  smoothness_worst 569 non-null    float64 
 27  compactness_worst 569 non-null    float64 
 28  concavity_worst  569 non-null    float64 
 29  concave_points_worst 569 non-null    float64 
 30  symmetry_worst   569 non-null    float64 
 31  fractal_dimension_worst 569 non-null    float64 
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
[16] dataset1["diagnosis"].describe()

```

DiseasePrediction.ipynb - Colab: + colab.research.google.com/drive/1oRvC2WjcvOtgX41WHh5Bn2GTHQNaUc#scrollTo=VD/RgjzfY0sa

```

File Edit View Insert Runtime Tools Help All changes saved
Files + Code + Text
dataset1["diagnosis"].describe()
   count    569
   unique     2
   top      'B'
   freq     357
Name: diagnosis, dtype: object

[17] # M - Malignant, B - Benign
dataset1["diagnosis"].unique()
array(['M', 'B'], dtype=object)

Exploratory Data Analysis

[18] y = dataset1["diagnosis"]
sns.countplot(y)

diag_temp = dataset1.diagnosis.value_counts()
print(diag_temp)

B    357
M    212
Name: diagnosis, dtype: int64

```

```

[21] print("Percentage of people having M form of cancer: " + str(round(diag_temp[1]*100/569, 2)))

```

DiseasePrediction.ipynb - Colab

```
[21] print("Percentage of people having M form of cancer: " + str(round(diag_temp[1]*100/569, 2)))
print("Percentage of people having B form of cancer: " + str(round(diag_temp[0]*100/569, 2)))
Percentage of people having M form of cancer: 37.26
Percentage of people having B form of cancer: 62.74

Splitting the dataset into training and testing

[22] from sklearn.model_selection import train_test_split
predictors = dataset1.drop("diagnosis", axis=1)
diagnosis = dataset1["diagnosis"]

[23] X_train, X_test, Y_train, Y_test = train_test_split(predictors, diagnosis, test_size=0.20, random_state=0)

[25] X_train.shape
(455, 31)

[26] X_test.shape
(114, 31)

[27] Y_train.shape
(455,)

[28] Y_test.shape
(114,)

Disk 66.03 GB available [29] from sklearn.metrics import accuracy_score
0s completed at 9:41 AM
```

DiseasePrediction.ipynb - Colab

```
[28] Y_test.shape
(114,)

[29] from sklearn.metrics import accuracy_score

Logistic Regression

[30] from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, Y_train)
Y_pred_lr = lr.predict(X_test)

[31] Y_pred_lr.shape
(114,)

[32] lr_score = round(accuracy_score(Y_pred_lr, Y_test)*100, 2)
print("The accuracy score of Logistic Regression is: " + str(lr_score) + "%")
The accuracy score of Logistic Regression is: 58.77%
```

Decision Tree Regression

```
[33] from sklearn.tree import DecisionTreeClassifier
max_accuracy = 0
for x in range(200):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train, Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
    if(current_accuracy > max_accuracy):
        max_accuracy = current_accuracy
    next_x = x
```

Disk 66.03 GB available 0s completed at 9:41 AM

DiseasePrediction.ipynb - Colab

File Edit View Insert Runtime Tools Help All changes saved

Decision Tree Regression

```
[33] from sklearn.tree import DecisionTreeClassifier
max_accuracy = 0

for x in range(2000):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train, Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
    if(current_accuracy > max_accuracy):
        max_accuracy = current_accuracy
        best_X = x

dt = DecisionTreeClassifier(random_state=best_X)
dt.fit(X_train, Y_train)
Y_pred_dt = dt.predict(X_test)

[34] Y_pred_dt.shape
(114,)

[35] dt.score = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
print("The accuracy score using Decision Tree is: " + str(dt.score) + "%")
The accuracy score using Decision Tree is: 93.86 %
```

Random Forest Regression

```
[36] from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0

for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train, Y_train)
```

Decision Tree Accuracy Score:

```
[35] dt.score = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
print("The accuracy score using Decision Tree is: " + str(dt.score) + "%")
The accuracy score using Decision Tree is: 93.86 %
```

Random Forest Regression Accuracy Score:

```
[36] from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0

for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train, Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf, Y_test)*100, 2)
    if(current_accuracy > max_accuracy):
        max_accuracy = current_accuracy
        best_X = x

rf = RandomForestClassifier(random_state=best_X)
rf.fit(X_train, Y_train)
Y_pred_rf = rf.predict(X_test)

[37] Y_pred_rf.shape
(114,)

[38] rf.score = round(accuracy_score(Y_pred_rf, Y_test)*100, 2)
print("The accuracy score using Random Forest Regressor is: " + str(rf.score) + "%")
The accuracy score using Random Forest Regressor is: 98.25 %
```

Diabetes

```
[39] type(dataset2)
pandas.core.frame.DataFrame
```

```

DiseasePrediction.ipynb - Colab: + https://colab.research.google.com/drive/1oRc2WjPzvOtgX41WHs5n2GfHQN4uc#scrollTo=VD7RjzsfY0sa
File Edit View Insert Runtime Tools Help All changes saved
Files + Code + Text Diabetes
[38] type(dataset2)
pandas.core.frame.DataFrame
[48] dataset2.shape
(768, 9)
[41] dataset2.head()
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
0 6 148 72 35 0 33.6 0.627 50 1
1 1 85 66 29 0 26.6 0.381 31 0
2 8 135 64 0 0 23.3 0.672 32 1
3 1 89 66 23 94 28.1 0.167 21 0
4 0 137 40 35 168 43.1 2.288 33 1

[50] dataset2.sample(5)
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
55 7 124 70 33 216 25.6 0.161 37 0
51 0 107 76 0 0 45.3 0.666 24 0
25 4 171 72 0 0 43.6 0.479 26 1
70 3 78 70 0 0 32.5 0.270 39 0
62 0 132 78 0 0 32.4 0.393 21 0

[43] dataset2.describe()
Disk 66.03 GB available Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
0s completed at 9:41 AM
09:44 IN 09/12/2021
DiseasePrediction.ipynb - Colab: + https://colab.research.google.com/drive/1oRc2WjPzvOtgX41WHs5n2GfHQN4uc#scrollTo=VD7RjzsfY0sa
File Edit View Insert Runtime Tools Help All changes saved
Files + Code + Text
[43] dataset2.describe()
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
count 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000
mean 3.845052 120.894531 69.105469 20.536458 79.799479 31.992576 0.471876 33.240885 0.348958
std 3.369978 31.972618 19.355907 15.952218 115.244002 7.884160 0.331329 11.760232 0.476951
min 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.078000 21.000000 0.000000
25% 1.000000 99.000000 62.000000 0.000000 27.300000 0.243750 24.000000 0.000000
50% 3.000000 117.000000 72.000000 23.000000 30.500000 32.000000 0.372500 29.000000 0.000000
75% 6.000000 140.250000 80.000000 32.000000 127.250000 36.600000 0.626250 41.000000 1.000000
max 17.000000 199.000000 122.000000 99.000000 846.000000 67.100000 2.420000 81.000000 1.000000

[44] dataset2.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Pregnancies    768 non-null   int64  
 1   Glucose       768 non-null   int64  
 2   BloodPressure 768 non-null   int64  
 3   SkinThickness 768 non-null   int64  
 4   Insulin        768 non-null   int64  
 5   BMI            768 non-null   float64 
 6   DiabetesPedigreeFunction 768 non-null   float64 
 7   Age            768 non-null   int64  
 8   Outcome        768 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB

[46] dataset2[["Outcome"]].describe()
Disk 66.03 GB available
count 768.000000
mean 0.168058
0s completed at 9:41 AM
09:44 IN 09/12/2021

```

DiseasePrediction.ipynb - Colab

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[46] dataset2["Outcome"].describe()
   count    768.000000
   mean     0.348958
   std      0.425212
   min     0.000000
   25%    0.000000
   50%    0.000000
   75%    1.000000
   max     1.000000
Name: Outcome, dtype: float64

[47] dataset2["Outcome"].unique()
array([0, 1])

Exploratory Data Analysis

[48] y = dataset2["Outcome"]
sns.countplot(y)

out_temp = dataset2.Outcome.value_counts()
print(out_temp)

0    500
1    268
Name: Outcome, dtype: int64


```

Disk 66.03 GB available 0s completed at 9:41 AM

DiseasePrediction.ipynb - Colab

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Exploratory Data Analysis

[49] y = dataset2["Outcome"]
sns.countplot(y)

out_temp = dataset2.Outcome.value_counts()
print(out_temp)

0    500
1    268
Name: Outcome, dtype: int64


```

[49] # 0 -> No Diabetes  
# 1 -> Has Diabetes

```

[50] print("Percentage of people having diabetes: " + str(round(out_temp[1]*100/768, 2)))
print("Percentage of people not having diabetes: " + str(round(out_temp[0]*100/768, 2)))

Percentage of people having diabetes: 34.9
Percentage of people not having diabetes: 65.1

Splitting the dataset into training and testing

```

```

[51] from sklearn.model_selection import train_test_split

```

Disk 66.03 GB available 0s completed at 9:41 AM

DiseasePrediction.ipynb - Colab

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Splitting the dataset into training and testing

[51] from sklearn.model_selection import train_test_split
predictors = dataset2.drop("Outcome", axis=1)
outcome = dataset2["Outcome"]

[52] X_train, X_test, Y_train, Y_test = train_test_split(predictors, outcome, test_size=0.20, random_state=0)

[53] X_train.shape
(614, 8)

[54] Y_train.shape
(614,)

[55] X_test.shape
(154, 8)

[56] Y_test.shape
(154,)

[57] from sklearn.metrics import accuracy_score

Logistic Regression

[58] from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, Y_train)
Y_pred_lr = lr.predict(X_test)

```

0s completed at 9:41 AM

DiseasePrediction.ipynb - Colab

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Logistic Regression

[58] from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, Y_train)
Y_pred_lr = lr.predict(X_test)

[59] Y_pred_lr.shape
(154,)

[60] lr.score = round(accuracy_score(Y_pred_lr, Y_test)*100, 2)
print("The accuracy score of Logistic Regression is: " + str(lr.score) + "%")
The accuracy score of Logistic Regression is: 82.47%
```

The accuracy score of Logistic Regression is: 82.47%

Decision Tree Regression

```

[61] from sklearn.tree import DecisionTreeClassifier
max_accuracy = 0

for x in range(200):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train, Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
    if(current_accuracy > max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

dt = DecisionTreeClassifier(random_state=best_x)
dt.fit(X_train, Y_train)
Y_pred_dt = dt.predict(X_test)

```

0s completed at 9:41 AM

DiseasePrediction.ipynb - Colab

```
+ [61]: from sklearn.tree import DecisionTreeClassifier
max_accuracy = 0

for x in range(200):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train, Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
    if(current_accuracy > max_accuracy):
        max_accuracy = current_accuracy
        best_X = x

dt = DecisionTreeClassifier(random_state=best_X)
dt.fit(X_train, Y_train)
Y_pred_dt = dt.predict(X_test)

[62]: Y_pred_dt.shape
(154,)

[63]: dt_score = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
print("The accuracy score using Decision Tree is: " + str(dt_score) + "%")
The accuracy score using Decision Tree is: 81.82 %
```

Decision Tree Regression

```
+ [64]: from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0

for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train, Y_train)
    Y_pred_rf = rf.predict(X_test)
```

Random Forest Regressor

```
+ [65]: Y_pred_rf.shape
(154,)

[66]: rf.score = round(accuracy_score(Y_pred_rf, Y_test)*100, 2)
print("The accuracy score using Random Forest Regressor is: " + str(rf_score) + "%")
The accuracy score using Random Forest Regressor is: 83.77 %
```

Random Forest Classifier

```
+ [67]: type(dataset3)
pandas.core.frame.DataFrame
```

```
+ [68]: dataset3.shape
(303, 14)
```

Heart

DiseasePrediction.ipynb - Colab

File Edit View Insert Runtime Tools Help All changes saved

Files

- sample.data
- cancer.csv
- diabetes.csv
- heart.csv

+ Code + Text

```
time accuracy score using random forest: Regression 150 0.77 4
```

Heart

```
[67] type(dataset3)
pandas.core.frame.DataFrame
```

```
[68] dataset3.shape
(303, 14)
```

```
[69] dataset3.head()
   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
0 63 1 3 145 233 1 0 150 0 2.3 0 0 1 1
1 37 1 2 130 250 0 1 167 0 3.5 0 0 2 1
2 41 0 1 130 204 0 0 172 0 1.4 2 0 2 1
3 56 1 1 120 236 0 1 178 0 0.8 2 0 2 1
4 57 0 0 120 354 0 1 163 1 0.6 2 0 2 1
```

```
[70] dataset3.sample(5)
   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
185 44 1 0 112 290 0 0 153 0 0.0 2 1 2 0
76 51 1 2 125 245 1 0 166 0 2.4 1 0 2 1
132 42 1 1 120 255 0 1 162 0 0.0 2 0 2 1
213 61 0 0 145 307 0 0 146 1 1.0 1 0 3 0
271 61 1 3 134 234 0 1 145 0 2.6 1 2 2 0
```

Disk 66.03 GB available [71] dataset3.describe()

```
[71] dataset3.describe()
   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
count 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000 303.000000
mean 54.396337 0.683168 0.956997 131.623762 245.624026 0.148515 0.528053 149.646065 0.326733 1.09604 1.399340 0.729373 2.313531 0.544554
std 9.082101 0.466011 1.032052 17.538143 51.830751 0.556198 0.525860 22.905161 0.469794 1.161075 0.616226 1.022696 0.612277 0.498835
min 29.000000 0.000000 0.000000 94.000000 126.000000 0.000000 0.000000 71.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
25% 47.500000 0.000000 0.000000 120.000000 211.000000 0.000000 0.000000 133.500000 0.000000 0.000000 1.000000 0.000000 2.000000 0.000000
50% 55.000000 1.000000 1.000000 130.000000 240.000000 0.000000 1.000000 153.000000 0.000000 0.800000 1.000000 0.000000 2.000000 1.000000
75% 61.000000 1.000000 2.000000 140.000000 274.500000 0.000000 1.000000 166.000000 1.000000 1.600000 2.000000 1.000000 3.000000 1.000000
max 77.000000 1.000000 3.000000 200.000000 564.000000 1.000000 2.000000 202.000000 1.000000 6.200000 2.000000 4.000000 3.000000 1.000000
```

Disk 66.03 GB available [72] dataset3.info()

```
[72] <class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total: 14 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   age     303 non-null   int64  
 1   sex     303 non-null   int64  
 2   cp      303 non-null   int64  
 3   trestbps 303 non-null  int64  
 4   chol    303 non-null   int64  
 5   fbs     303 non-null   int64  
 6   restecg 303 non-null   int64  
 7   thalach 303 non-null   int64  
 8   exang   303 non-null   int64  
 9   oldpeak 303 non-null   float64 
 10  slope   303 non-null   int64  
 11  ca      303 non-null   int64  
 12  thal    303 non-null   int64  
 13  target  303 non-null   int64 
```

0s completed at 9:41 AM

DiseasePrediction.ipynb - Colab

```
[72]: dataset3.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 383 entries, 0 to 382
Data columns (total: 13):
 #   Column      Non-Null Count  Dtype  
 --- 
  0   age         383 non-null    int64  
  1   sex         383 non-null    int64  
  2   cp          383 non-null    int64  
  3   trestbps   383 non-null    int64  
  4   chol        383 non-null    int64  
  5   fbs         383 non-null    int64  
  6   restecg    383 non-null    int64  
  7   thalach    383 non-null    int64  
  8   exang       383 non-null    int64  
  9   oldpeak    383 non-null    float64 
  10  slope       383 non-null    int64  
  11  ca          383 non-null    int64  
  12  thal        383 non-null    int64  
  13  target      383 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB

[73]: dataset3["target"].describe()
count    383.000000
mean     0.544554
std      0.498835
min     0.000000
25%    0.000000
50%    1.000000
75%    1.000000
max     1.000000
Name: target, dtype: float64

[74]: dataset3["target"].unique()
array([1, 0])

Exploratory Data Analysis
```

DiseasePrediction.ipynb - Colab

```
[75]: dataset3["target"].unique()
array([1, 0])

Exploratory Data Analysis

[76]: y = dataset3["target"]
sns.countplot(y)

target_temp = dataset3.target.value_counts()
print(target_temp)

1    165
0    118
Name: target, dtype: int64

```

[77]: # 0 -> No heart disease  
# 1 -> Has heart disease

[78]: print("Percentage of people having heart disease: " + str(round(target\_temp[1]\*100/383, 2)))  
print("Percentage of people not having heart disease: " + str(round(target\_temp[0]\*100/383, 2)))

Percentage of people having heart disease: 54.46  
Percentage of people not having heart disease: 45.54

Splitting the dataset into training and testing

DiseasePrediction.ipynb - Colab

File Edit View Insert Runtime Tools Help All changes saved

Files

- sample.data
- cancer.csv
- diabetes.csv
- heart.csv

+ Code + Text

```
Percentage of people having heart disease: 54.46  
Percentage of people not having heart disease: 45.54  
  
Splitting the dataset into training and testing
```

```
[81] from sklearn.model_selection import train_test_split  
predictors = dataset.drop("target", axis=1)  
target = dataset[["target"]]  
  
[82] X_train, X_test, Y_train, Y_test = train_test_split(predictors, target, test_size=0.20, random_state=0)  
  
[85] X_train.shape  
(242, 13)  
  
[86] Y_train.shape  
(242,)  
  
[87] X_test.shape  
(61, 13)  
  
[88] Y_test.shape  
(61,)  
  
[89] from sklearn.metrics import accuracy_score  
  
Logistic Regression
```

```
[90] from sklearn.linear_model import LogisticRegression  
lr = LogisticRegression()  
  
[91] Y_pred_lr.shape  
(61,)  
  
[92] lr_score = round(accuracy_score(Y_pred_lr, Y_test)*100, 2)  
print("accuracy score of logistic Regression is: " + str(lr_score) + "%")  
The accuracy score of Logistic Regression is: 85.25%
```

Decision Tree Regression

```
[93] from sklearn.tree import DecisionTreeClassifier  
max_accuracy = 0  
for x in range(200):  
    dt = DecisionTreeClassifier(random_state=x)  
    dt.fit(X_train, Y_train)  
    Y_pred_dt = dt.predict(X_test)  
    current_accuracy = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)  
    if(current_accuracy > max_accuracy):  
        max_accuracy = current_accuracy  
        best_x = x  
  
dt = DecisionTreeClassifier(random_state=best_x)  
dt.fit(X_train, Y_train)  
Y_pred_dt = dt.predict(X_test)
```

Disk 66.03 GB available

0s completed at 9:41 AM

ENG IN 09:46 05/12/2021

DiseasePrediction.ipynb - Colab

File Edit View Insert Runtime Tools Help All changes saved

Files

- sample.data
- cancer.csv
- diabetes.csv
- heart.csv

+ Code + Text

```
Logistic Regression
```

```
[90] from sklearn.linear_model import LogisticRegression  
lr = LogisticRegression()  
  
lr.fit(X_train, Y_train)  
Y_pred_lr = lr.predict(X_test)  
  
[91] Y_pred_lr.shape  
(61,)  
  
[92] lr_score = round(accuracy_score(Y_pred_lr, Y_test)*100, 2)  
print("accuracy score of logistic Regression is: " + str(lr_score) + "%")  
The accuracy score of Logistic Regression is: 85.25%
```

Decision Tree Regression

```
[93] from sklearn.tree import DecisionTreeClassifier  
max_accuracy = 0  
for x in range(200):  
    dt = DecisionTreeClassifier(random_state=x)  
    dt.fit(X_train, Y_train)  
    Y_pred_dt = dt.predict(X_test)  
    current_accuracy = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)  
    if(current_accuracy > max_accuracy):  
        max_accuracy = current_accuracy  
        best_x = x  
  
dt = DecisionTreeClassifier(random_state=best_x)  
dt.fit(X_train, Y_train)  
Y_pred_dt = dt.predict(X_test)
```

Disk 66.03 GB available

0s completed at 9:41 AM

ENG IN 09:46 05/12/2021

The accuracy score of Logistic Regression is: 85.25%

Decision Tree Regression

```
[93]: from sklearn.tree import DecisionTreeClassifier
max_accuracy = 0
for x in range(2000):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train, Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
    if(current_accuracy > max_accuracy):
        max_accuracy = current_accuracy
        best_x = x
dt = DecisionTreeClassifier(random_state=best_x)
dt.fit(X_train, Y_train)
Y_pred_dt = dt.predict(X_test)
```

[94]: Y\_pred\_dt.shape  
(61,)

```
[95]: dt.score = round(accuracy_score(Y_pred_dt, Y_test)*100, 2)
print("The accuracy score using Decision Tree is: " + str(dt.score) + "%")  
The accuracy score using Decision Tree is: 81.97 %
```

Random Forest Regression

```
[96]: from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0
for x in range(2000):
```

The accuracy score using Decision Tree is: 81.97 %

Random Forest Regression

```
[96]: from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0
for x in range(2000):
```

```
[96]: from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0
for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train, Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf, Y_test)*100, 2)
    if(current_accuracy > max_accuracy):
        max_accuracy = current_accuracy
        best_x = x
rf = RandomForestClassifier(random_state=best_x)
rf.fit(X_train, Y_train)
Y_pred_rf = rf.predict(X_test)
```

[97]: Y\_pred\_rf.shape  
(61,)

```
[97]: rf.score = round(accuracy_score(Y_pred_rf, Y_test)*100, 2)
print("The accuracy score using Random Forest Regressor is: " + str(rf.score) + "%")  
The accuracy score using Random Forest Regressor is: 90.16 %
```

## Website Output:

← → ⌂ 127.0.0.1:5000

Disease Prediction Home About Cancer Diabetes Heart

## Details

Welcome to the home page of the app.

This is mainly based on as the application of the machine learning, meant to be employed in the remote and the downtrodden area.

## Overview

In today's time we see a lot of the shortage of the doctors in the world especially in India. A lot of people are suffering a lot without the help of the proper medical checkup. Also most of the cases many cases arise leading to death due to lack of timely medical checkup.

So to cope up with all of those problems this app is designed which would prove its benefits upto much extent.

---

## Application

- To remove the dependencies on the doctors
- To help out the poor and helpless people with the normal medical checkup
- To help people avoid paying huge amount to the doctors unnecessarily
- To extend the role of the technology in the medical field

## Creator :

Smrithi Agrawal & Karen Castelino

Disease Prediction    Home    About    Cancer    Diabetes    Heart

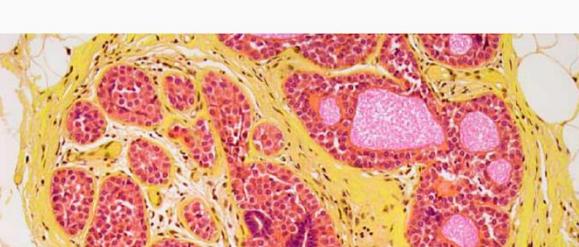
## Details :

**This is a basic machine learning and deep learning based medical app.**

The prediction about the disease is made with the help of the machine learning model which itself is trained on the thousands of the datasets

So with the following paragraph I gonna be describing about the types of the diseases which gonna be getting employed in this app.

### 1. Breast Cancer



← → ⌂ ① 127.0.0.1:5000/about

Disease Prediction Home About Cancer Diabetes Heart

Breast cancer is cancer that develops from breast tissue. Signs of breast cancer may include a lump in the breast, a change in breast shape, dimpling of the skin, fluid coming from the nipple, a newly inverted nipple, or a red or scaly patch of skin.

## Symptoms

Signs and symptoms of breast cancer may include:

- A breast lump or thickening that feels different from the surrounding tissue
- Change in the size, shape or appearance of a breast
- Changes to the skin over the breast, such as dimpling
- A newly inverted nipple
- Peeling, scaling, crusting or flaking of the pigmented area of skin surrounding the nipple (areola) or breast skin
- Redness or pitting of the skin over your breast, like the skin of an orange

## Causes

Doctors know that breast cancer occurs when some breast cells begin to grow abnormally. These cells divide more rapidly than healthy cells do and continue to accumulate, forming a lump or mass. Cells may spread (metastasize) through your breast to your lymph nodes or to other parts of your body. Breast cancer most often begins with cells in the milk-producing ducts (invasive ductal carcinoma). Breast cancer may also begin in the glandular tissue called lobules (invasive lobular carcinoma) or in other cells or tissue within the breast. Researchers have identified hormonal, lifestyle and environmental factors that may increase your risk of breast cancer. But it's not clear why some people who have no risk factors develop cancer, yet other people with risk factors never do. It's likely that breast cancer is caused by a complex interaction of your genetic makeup and your environment.

← → ⌂ ① 127.0.0.1:5000/about

Disease Prediction Home About Cancer Diabetes Heart

## 2. Diabetes

### Diabetes: Complications

The diagram illustrates the various complications of diabetes, divided into two main categories: Macrovascular and Microvascular.

- Macrovascular:**
  - Stroke
  - Heart disease and hypertension
  - Peripheral vascular disease
  - Foot problems
- Microvascular:**
  - Diabetic eye disease (retinopathy and cataracts)
  - Renal disease
  - Neuropathy
  - Foot problems

Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food you eat. Insulin, a hormone made by the pancreas, helps glucose from food get into your cells to be used for energy.

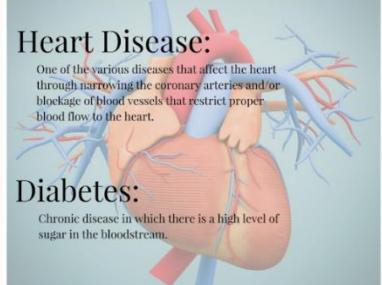
### What health problems can people with diabetes develop?

Over time, high blood glucose leads to problems such as

← → ⌂ 127.0.0.1:5000/about

Disease Prediction Home About Cancer Diabetes Heart

### 3.Heart Disease



**Heart Disease:**  
One of the various diseases that affect the heart through narrowing the coronary arteries and/or blockage of blood vessels that restrict proper blood flow to the heart.

**Diabetes:**  
Chronic disease in which there is a high level of sugar in the bloodstream.

Excess sugar that cannot fit in the body's cells accumulates in the bloodstream, tissues and organs of the body.

Elevated blood sugar levels cause type I and type II diabetes and heart disease, along with heart attacks.

Having high amounts of sugar in the bloodstream develops into fat that clogs the blood vessels, narrows, inflames and causes them to wear and tear.

HEART DISEASES WELLNESS INSTITUTE

**Overview**

Heart disease describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease; heart rhythm problems (arrhythmias); and heart defects you're born with (congenital heart defects), among others. The term "heart disease" is often used interchangeably with the term "cardiovascular disease." Cardiovascular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke. Other heart conditions, such as those that affect your heart's muscle, valves or rhythm, also are considered forms of heart disease.

Many forms of heart disease can be prevented or treated with healthy lifestyle choices

### Final Output:

Cancer.html

Disease Prediction Home About Cancer Diabetes Heart

### Details :

#### Welcome

Radius\_mean :

Texture\_mean :

Perimeter\_mean :

Area\_mean :

Smoothness\_mean :

Compactness\_mean:

Concavity\_mean:

Concave points\_mean:

Symmetry\_mean:

Fractal\_dimension\_mean:

Radius\_se:

Texture\_se:

← → ⌛ 127.0.0.1:5000/result

Disease Prediction Home About Cancer Diabetes Heart

The result is :

**Sorry ! Suffering**

Want to know more about the Disease and its symptoms ?

[Click Me !](#)

Diabetes.html

Disease Prediction Home About Cancer Diabetes Heart

**Details Diabetes:**

**Welcome**

**Enter the Following parameters :**

Pregnancies :

Glucose :

BloodPressure :

SkinThickness :

Insulin :

BMI :

DiabetesPedigreeFunction :

Age :

The result is :

Congrats ! you are Healthy

Want to know more about the Disease and its symptoms ?

[Click Me !](#)

Heart.html

**Details Heart:**

**Welcome**

**Enter the Following parameters :**

Age :

Sex(1:Male,0:Female) :

Chest pain type :

Trestbps :

Serum cholestorol in mg/dl :

Restecg :

Thalach :

Exang :

Oldpeak :

Slope :

Thal :

The result is :

**Sorry ! Suffering**

Want to know more about the Disease and its symptoms ?

[Click Me !](#)

### **Conclusion:**

The methods logistic regression, decision tree and random forest are analyzed for predicting the type of treatment that can be offered for the patient with cancer, diabetes or heart disease.

The success rate of classification for cancer is 58.77% obtained by Logistic Regression. The accuracy using Decision Tree Regression is 93.86%. The prediction percentage using Random Forest Regression is 98.25%.

The success rate of classification for Diabetes is 82.47% obtained by Logistic Regression. The accuracy using Decision Tree Regression is 81.82%. The prediction percentage using Random Forest Regression is 83.77%.

The success rate of classification for Heart Disease is 85.25% obtained by Logistic Regression. The accuracy using Decision Tree regression is 81.97%. The prediction percentage using Random Forest Regression is 90.16%.

### **Future Scope**

As a future enhancement, how the preprocessing of dataset can be improvised for increasing the success rate in classification and prediction process. Implementation of various other diseases can be done by the use of different algorithms to achieve better accuracy (most probably neural networks). An improved and user-friendly front-end can be useful for better accessibility to the website.

### **Limitations**

- All the necessary input values must be present and accurate.
- The accuracy is not as good as we expect it to be, and sometimes this might result in a wrong prediction.
- The prediction might change after some time depending on the input attributes

### **References:**

<https://www.geeksforgeeks.org/random-forest-regression-in-python/>

<https://ieeexplore.ieee.org/document/8455058>

<https://ieeexplore.ieee.org/document/9368899>

<https://www.sciencedirect.com/science/article/pii/S1319157820304134>

