Guru Nanak Dev University

Amritsar

Department Of Electronics Technology

A project based Report

Multiple Disease Detection

Submitted By-

Aman Kumar

Semester – VI (ECE)

Roll number -17042011210

Submitted To –

Dr. Sukhdeep kaur

Table Of Contents

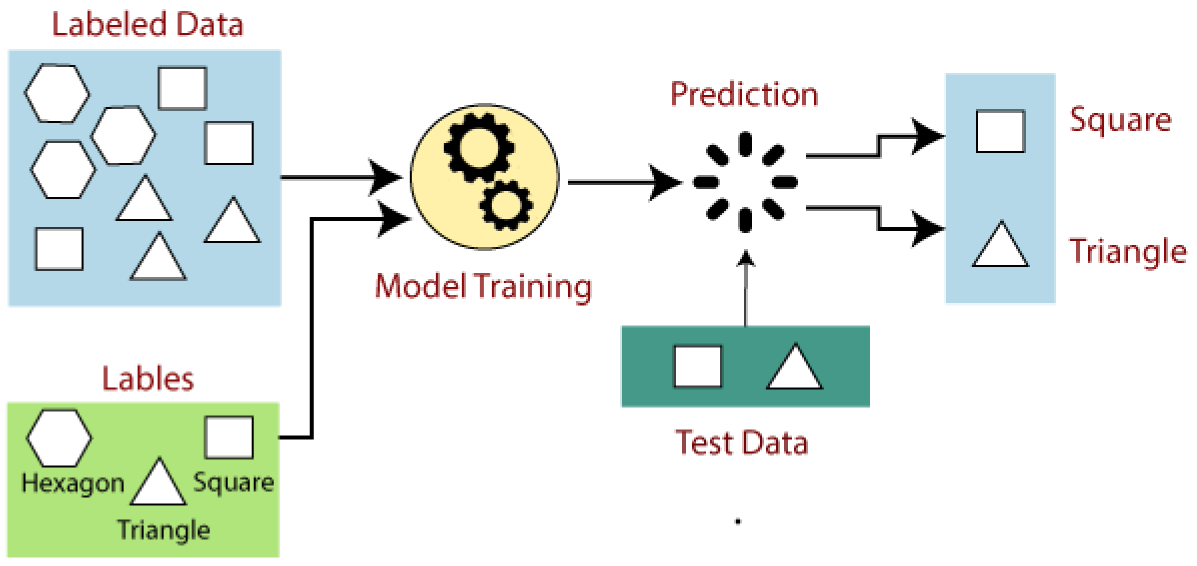
INTRODUCTION

Machine Learning Approach for Identifying Multiple Disease Prediction Using Machine Learning is based on prediction modelling that predicts disease of the patients according to the Laboratory test measures provided by the users as an i/p to the system. This report gives an idea of predicting multiple diseases using Machine Learning algorithms. Here we will use the concept of supervised Machine Learning in which implementation will be done by applying Support Vector Machine (SVM) , Logistic Regression algorithms which will help in early prediction of diseases accurately and better patients care. The results ensured that the system would be functional and user oriented for patients for timely diagnoses of diseases in a patient.

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.



Objective

1. Our main aim is to provide a quick medical diagnosis to the patients living in rural area.
2. In present days it is very useful for post covid contactless system in rural health service .
3. The goal is to provide access to medical specialists e-Doctor.
4. This system enhance quality of health care
5. This project helps in saving the money which we give for appointments to Doctors.
6. It help a lot of people else one monitor the persons condition and take the necessary precaution thus increasing the life expectancy

Software required

1. Google Colab Notebook
2. Anaconda.
3. Pycharm.

Supervised Learning Algorithm

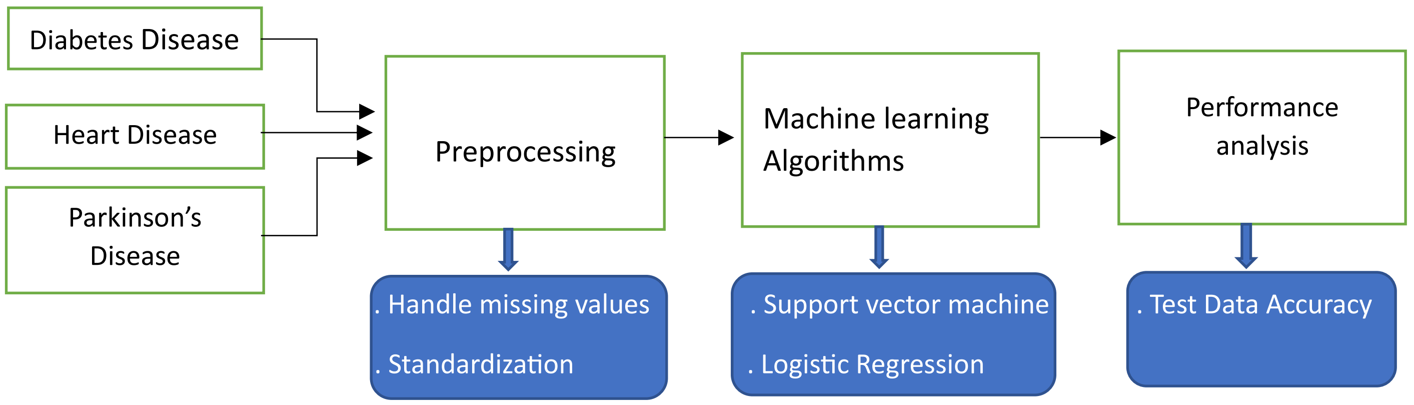
1. Support Vector Machine (SVM).

   It can classifies both linear and non linear data . it first map each data item into and n-dimentional features space where n is the number of features. It then identify the hyperplane that separates the data items into two classes while maximising the marginal distance for both classes and minimizing the classification errors.

2. Logistic Regression

   Logistic regression is a powerful and well establish method for supervised classification. It can be considers as an extension of ordinary regression and can model only a dichotomous of an event . LR helps in finding the probability that a new instance belongs to a certain class. Since it is a probability ,the outcome lies between 0 and 1.
   This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.

Block diagram



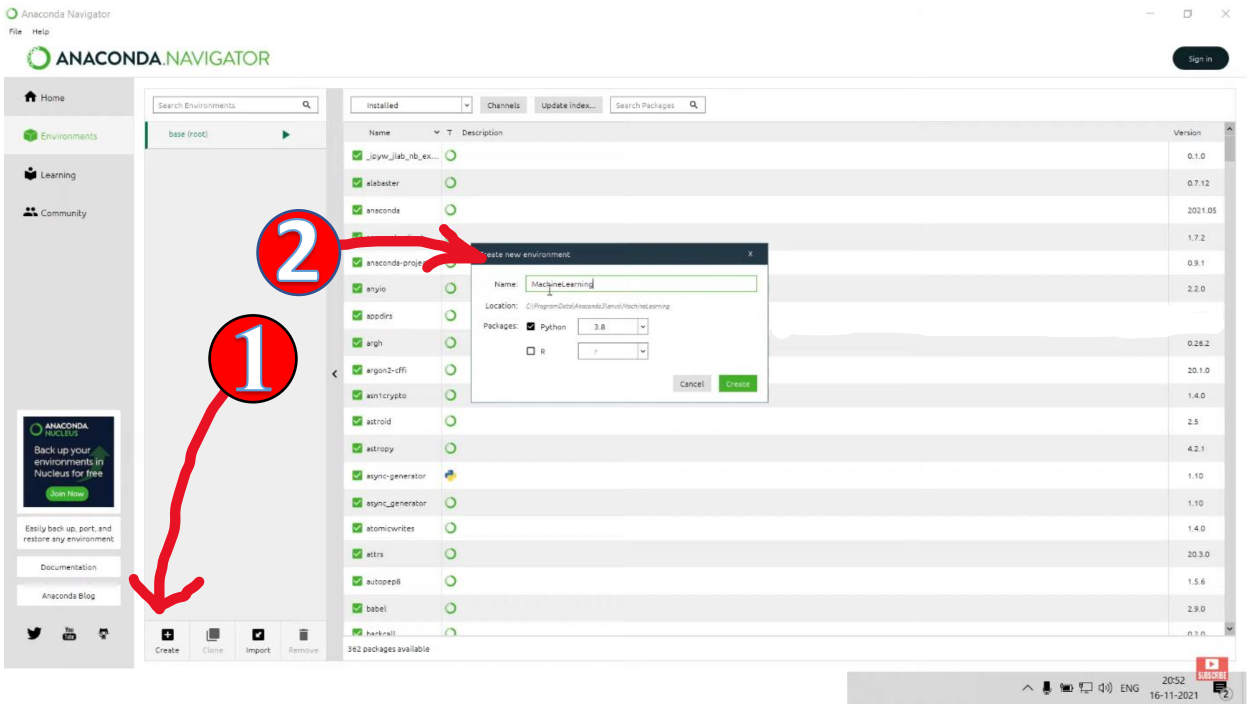
Working

So, before execution we have some pre-requisites that we need to download or install i.e., anaconda environment, Pycharm and a code editor (Google colab Notebook) .
Anaconda: Anaconda is like a package of libraries and offers a great deal of information which allows a data engineer to create multiple environments and install required libraries easy and neat.
PyCharm : PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.
Google colab notebook: Google colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Install the prerequisites mentioned above.

Step1 – Set Environment in Anaconda
Open anaconda prompt and create a new environment. To create an environment use the commands given below. Replace MachineLearning by the name of environment you want to give.

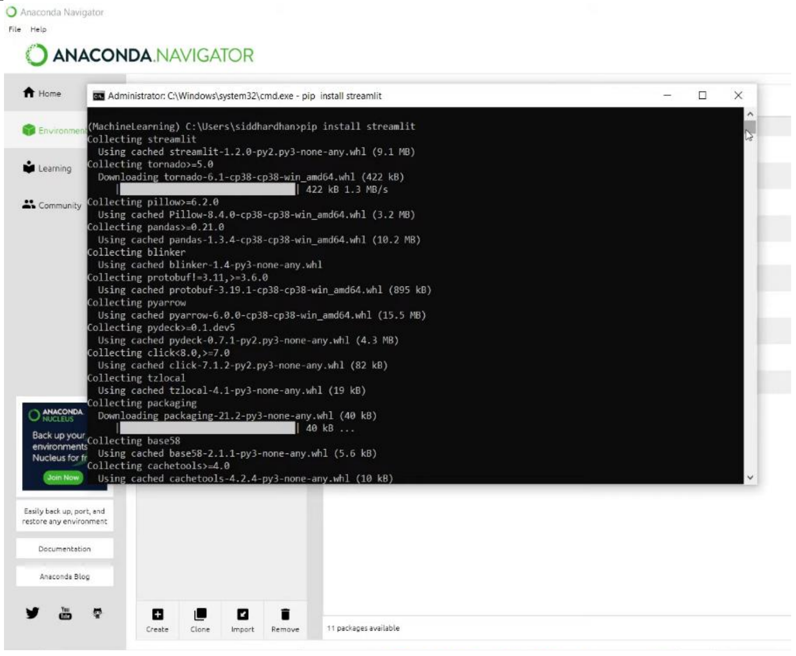
- conda create -n "MachineLearning"
- conda activate "MachineLearning"



Step2

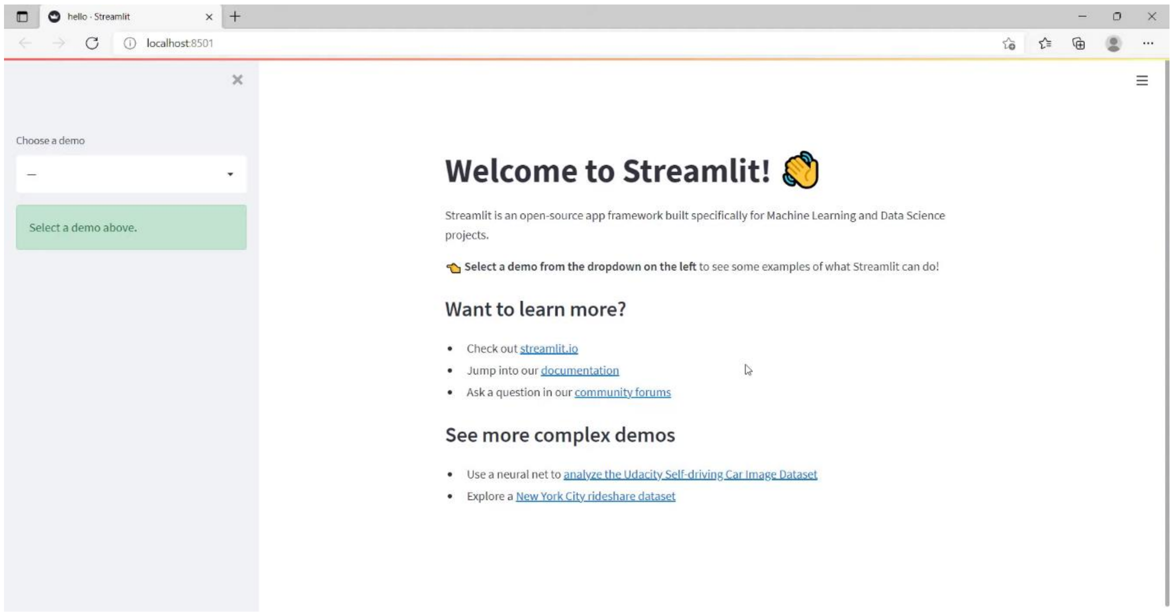
open the terminal of our above environment and install the package Streamlit using below command

- pip install streamlit


ANACONDA.NAVIGATOR
Administrator: C:\Windows\system32\cmd.exe - pip install streamlit

```
(MachineLearning) C:\Users\siddhardhan>pip install streamlit
Collecting streamlit
  Using cached streamlit-1.2.0-py2.py3-none-any.whl (9.1 MB)
Collecting tornado>=5.0
  Downloading tornado-6.1-cp38-cp38-win_amd64.whl (422 kB)
     |████████████████████████████████| 422 kB 1.3 MB/s
Collecting pillow>=6.2.0
  Using cached Pillow-8.4.0-cp38-cp38-win_amd64.whl (3.2 MB)
Collecting pandas>=0.21.0
  Using cached pandas-1.3.4-cp38-cp38-win_amd64.whl (10.2 MB)
Collecting blinker
  Using cached blinker-1.4-py3-none-any.whl
Collecting protobuf!=3.11,>=3.6.0
  Using cached protobuf-3.19.1-cp38-cp38-win_amd64.whl (895 kB)
Collecting pyarrow
  Using cached pyarrow-6.0.0-cp38-cp38-win_amd64.whl (15.5 MB)
Collecting pydeck>=0.1.dev5
  Using cached pydeck-0.7.1-py2.py3-none-any.whl (4.3 MB)
Collecting click<8.0,>=7.0
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting tzlocal
  Using cached tzlocal-4.1-py3-none-any.whl (19 kB)
Collecting packaging
  Downloading packaging-21.2-py3-none-any.whl (40 kB)
     |████████████████████████████████| 40 kB ...
Collecting base58
  Using cached base58-2.1.1-py3-none-any.whl (5.6 kB)
Collecting cachetools>=4.0
  Using cached cachetools-4.2.4-py3-none-any.whl (10 kB)
```

After that we will be able to see the default browser of streamlit



Welcome to Streamlit! 👋

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects.

👈 **Select a demo from the dropdown on the left** to see some examples of what Streamlit can do!

Want to learn more?

- Check out streamlit.io
- Jump into our documentation
- Ask a question in our community forums

See more complex demos

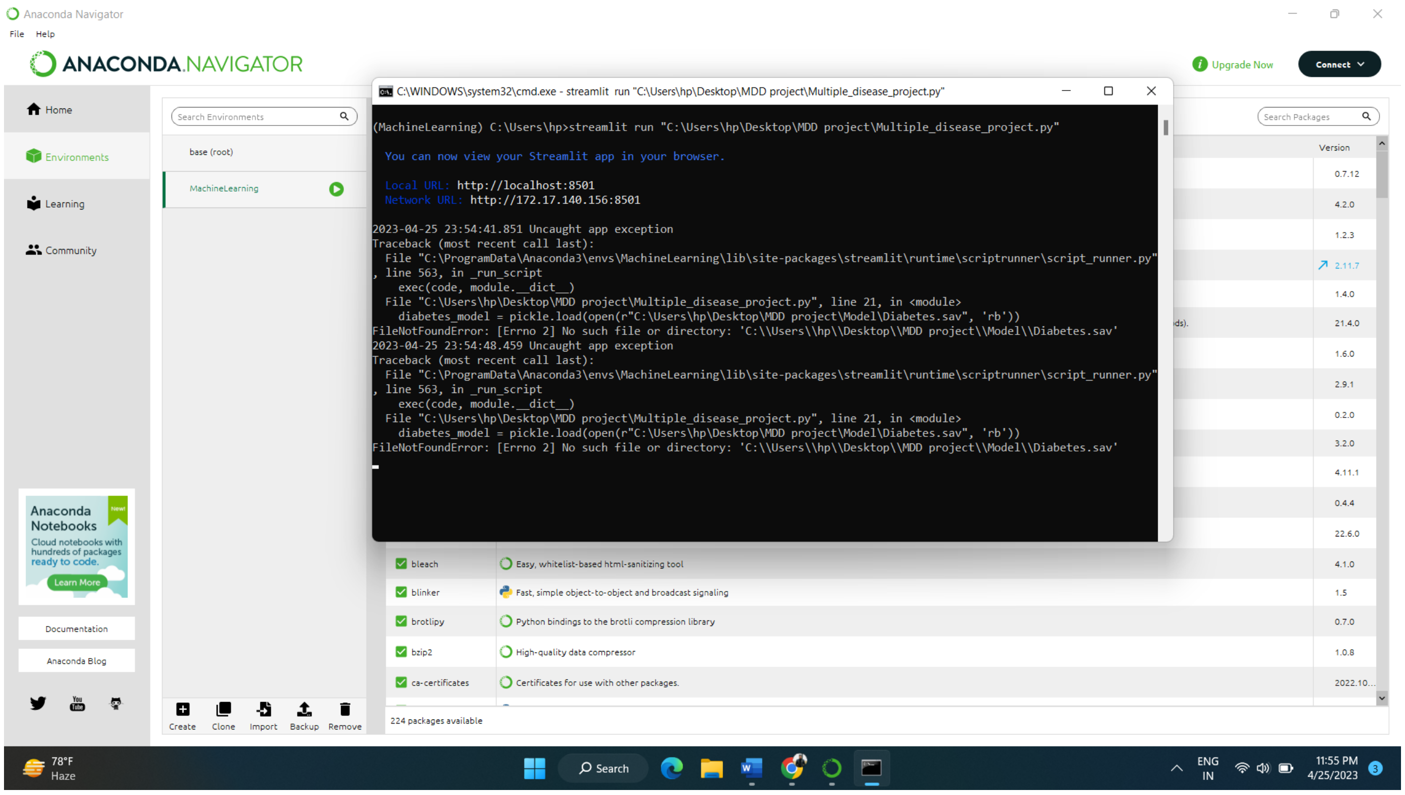
- Use a neural net to analyze the Udacity Self-driving Car Image Dataset
- Explore a New York City rideshare dataset

Step3

To run the code, type "streamlit run "path/filename.extension" in our case *streamlit run "D:\Exicuted Projects\Multiple Disease Prediction System\multiple_disease_pred.py"*

Open the environment terminal and run the code by above command .

Here we need to give the address of python code file which in our case is pycharm file.



C:\WINDOWS\system32\cmd.exe - streamlit run "C:\Users\hp\Desktop\MDD project\Multiple_disease_project.py"

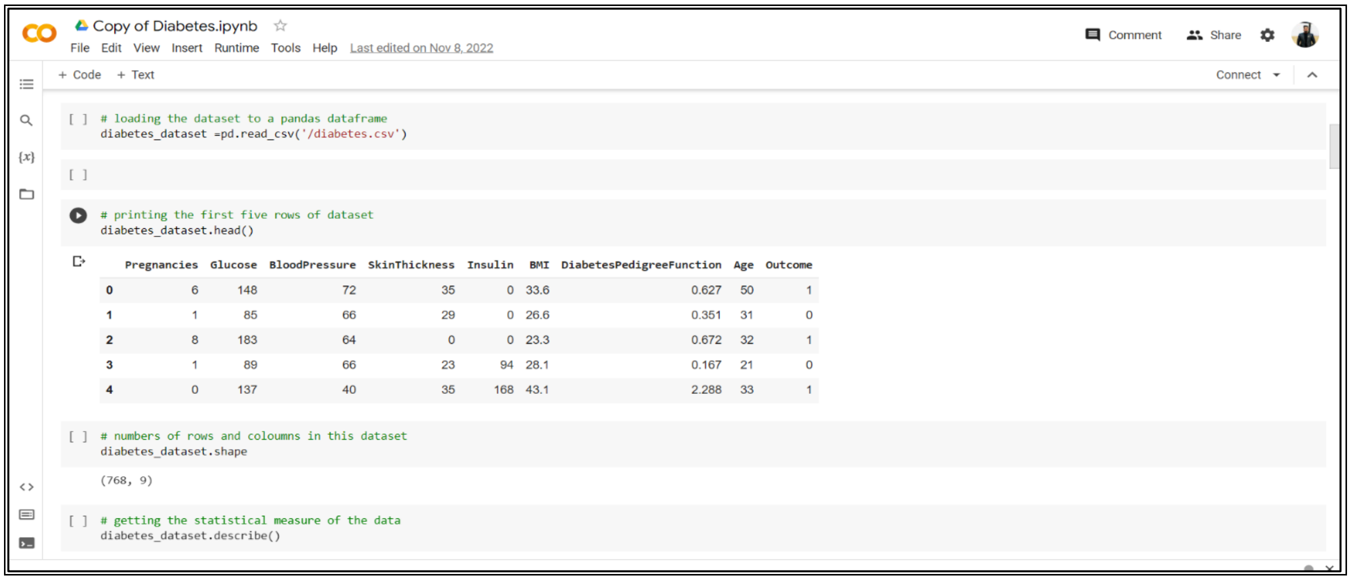
```
(MachineLearning) C:\Users\hp>streamlit run "C:\Users\hp\Desktop\MDD project\Multiple_disease_project.py"

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.17.140.156:8501

2023-04-25 23:54:41.851 Uncaught app exception
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\envs\MachineLearning\lib\site-packages\streamlit\runtime\scriptrunner\script_runner.py"
, line 563, in _run_script
    exec(code, module.__dict__)
  File "C:\Users\hp\Desktop\MDD project\Multiple_disease_project.py", line 21, in <module>
    diabetes_model = pickle.load(open(r"C:\Users\hp\Desktop\MDD project\Model\Diabetes.sav", 'rb'))
FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\hp\\Desktop\\MDD project\\Model\\Diabetes.sav'
2023-04-25 23:54:48.459 Uncaught app exception
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\envs\MachineLearning\lib\site-packages\streamlit\runtime\scriptrunner\script_runner.py"
, line 563, in _run_script
    exec(code, module.__dict__)
  File "C:\Users\hp\Desktop\MDD project\Multiple_disease_project.py", line 21, in <module>
    diabetes_model = pickle.load(open(r"C:\Users\hp\Desktop\MDD project\Model\Diabetes.sav", 'rb'))
FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\hp\\Desktop\\MDD project\\Model\\Diabetes.sav'
```

		Version
✓ bleach	Easy, whitelist-based html-sanitizing tool	4.1.0
✓ blinker	Fast, simple object-to-object and broadcast signaling	1.5
✓ brotlipy	Python bindings to the brotli compression library	0.7.0
✓ bzip2	High-quality data compressor	1.0.8
✓ ca-certificates	Certificates for use with other packages.	2022.10...

Data Description

In this project we will be doing multiple disease prediction the diseases are Diabetes, Heart disease and Parkinsons and Stroke and all the data sets used are tabular data .

- **Diabetes dataset**

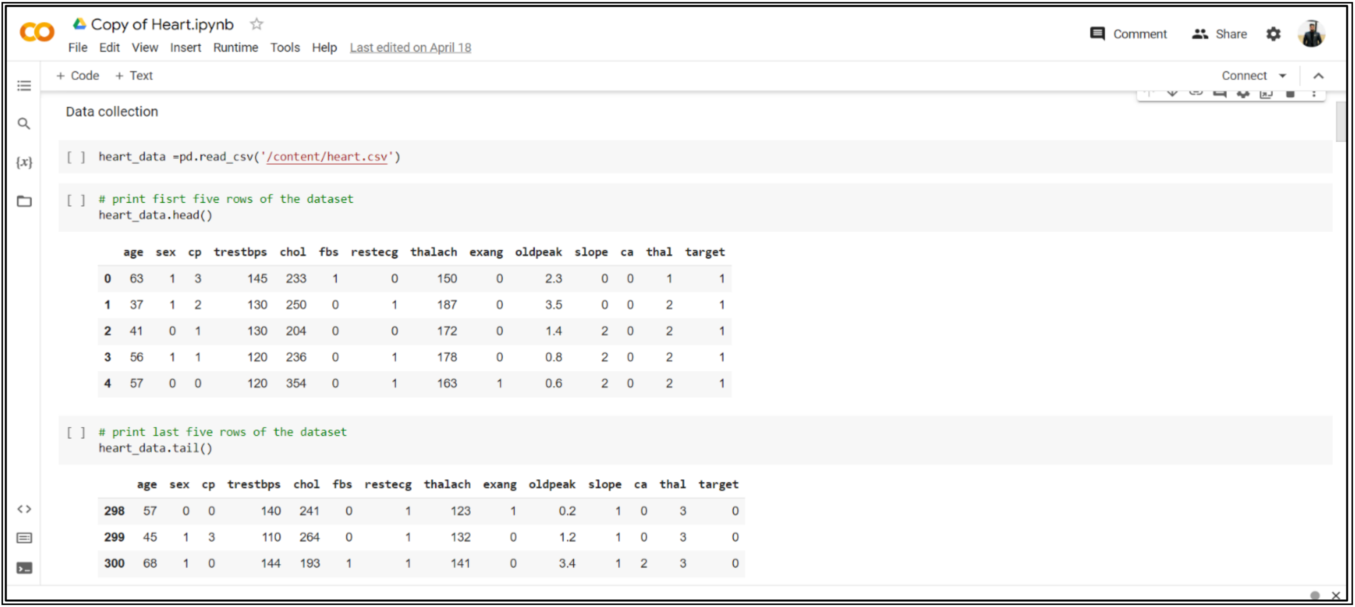

Copy of Diabetes.ipynb
File Edit View Insert Runtime Tools Help Last edited on Nov 8, 2022
Comment Share
+ Code + Text
Connect

```
[ ] # loading the dataset to a pandas dataframe
    diabetes_dataset =pd.read_csv('/diabetes.csv')

[ ]

    # printing the first five rows of dataset
    diabetes_dataset.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	0.351	31	0
2	8	183	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

```
[ ] # numbers of rows and coloumns in this dataset
    diabetes_dataset.shape

    (768, 9)

[ ] # getting the statistical measure of the data
    diabetes_dataset.describe()
```

- **Heart Disease Dataset**


Copy of Heart.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 18
Comment Share
+ Code + Text
Connect

Data collection

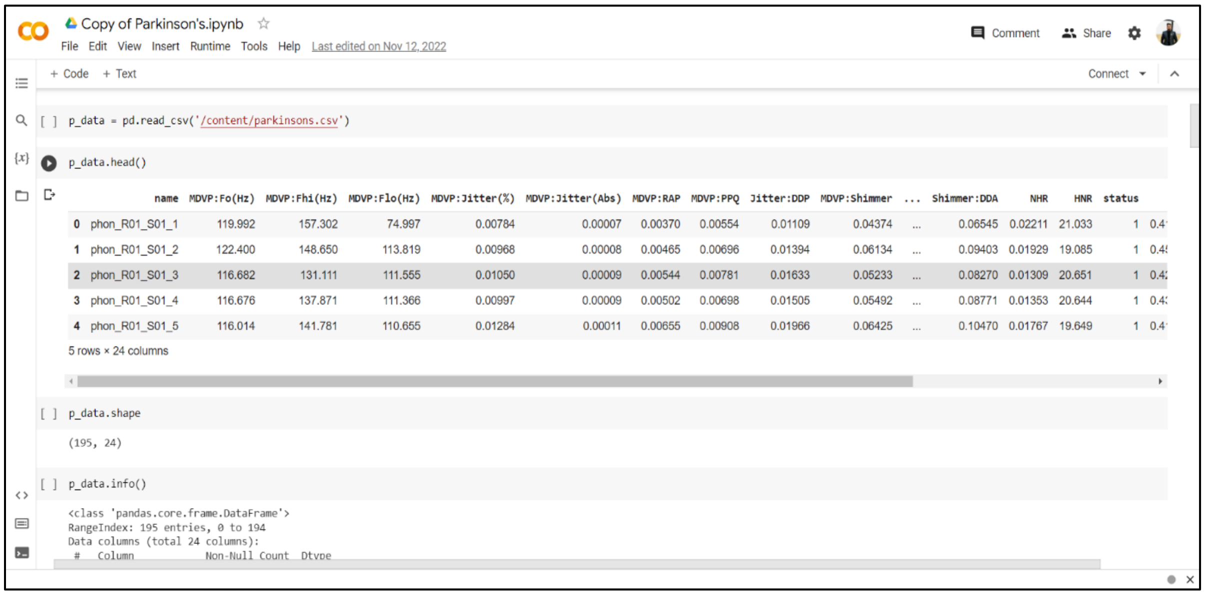
```
[ ] heart_data =pd.read_csv('/content/heart.csv')

[ ] # print fisrt five rows of the dataset
    heart_data.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	187	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

```
[ ] # print last five rows of the dataset
    heart_data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0

- **Parkinson disease dataset**


Copy of Parkinson's.ipynb
File Edit View Insert Runtime Tools Help Last edited on Nov 12, 2022
Comment Share
+ Code + Text
Connect

```
p_data = pd.read_csv('/content/parkinsons.csv')
```

```
p_data.head()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1

5 rows × 24 columns

```
p_data.shape
```

```
(195, 24)
```

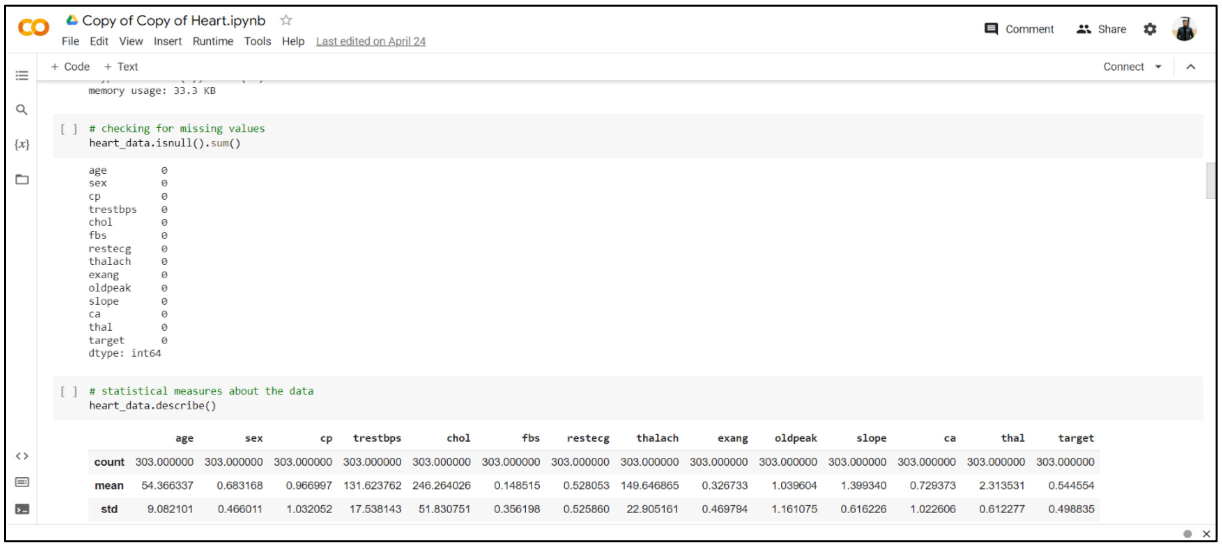
```
p_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
```

Preprocessing the data

Check missing values in all the disease datasets . if dataset consist of missing values .we must handle those missing values

- dataset.isnull().sum()


Copy of Copy of Heart.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 24
Comment Share
+ Code + Text
Connect

```
memory usage: 33.3 KB
```

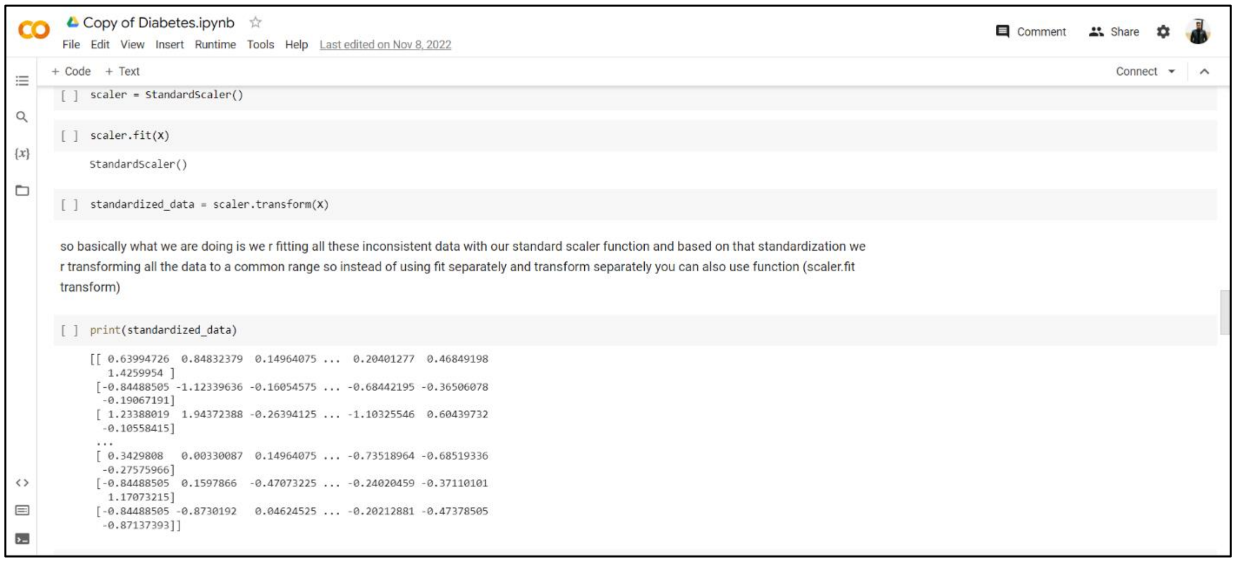
```
# checking for missing values
heart_data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
# statistical measures about the data
heart_data.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	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835

- Standardize the data


Copy of Diabetes.ipynb
File Edit View Insert Runtime Tools Help Last edited on Nov 8, 2022
Comment Share
+ Code + Text
Connect

```
[ ] scaler = StandardScaler()

[ ] scaler.fit(X)

    StandardScaler()

[ ] standardized_data = scaler.transform(X)
```

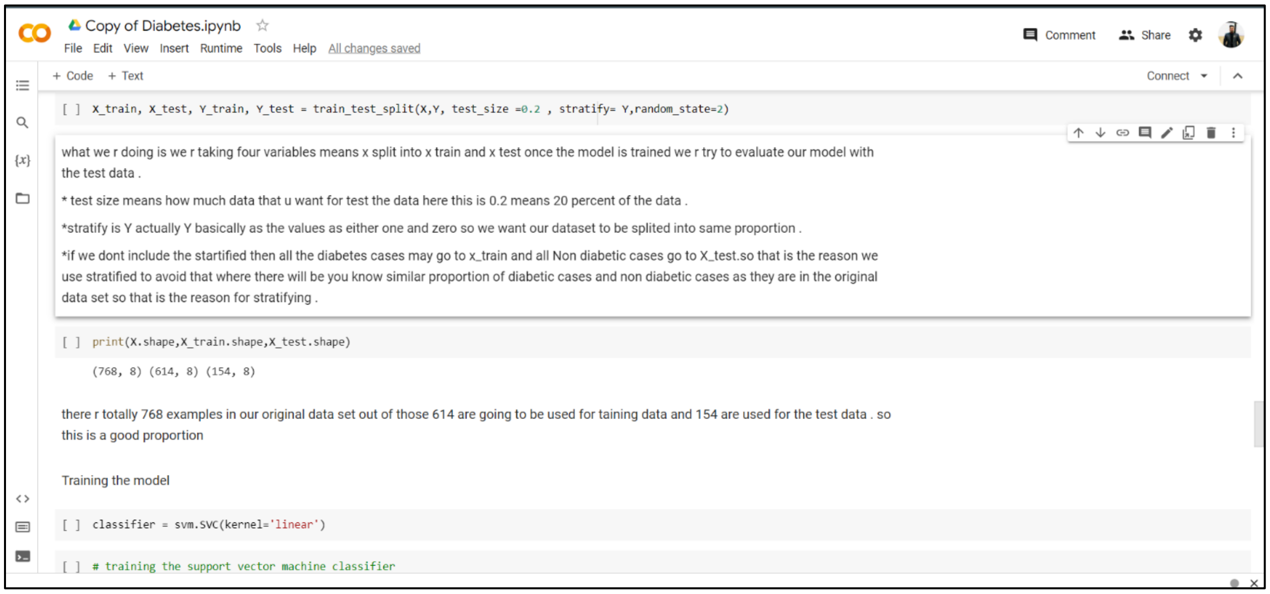
so basically what we are doing is we r fitting all these inconsistent data with our standard scaler function and based on that standardization we r transforming all the data to a common range so instead of using fit separately and transform separately you can also use function (scaler.fit transform)

```
[ ] print(standardized_data)

    [[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
       1.4259954 ]
     [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
      -0.19067191]
     [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
      -0.10558415]
     ...
     [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
      -0.27575966]
     [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
       1.17073215]
     [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
      -0.87137393]]
```

Splitting the data (Training and Testing)

what we are doing is we are taking four variables means x split into x train and x test once the model is trained we are try to evaluate our model with the test data .

- `X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =0.2 , stratify= Y,random_state=2)`



```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =0.2 , stratify= Y,random_state=2)
```

what we r doing is we r taking four variables means x split into x train and x test once the model is trained we r try to evaluate our model with the test data .

* test size means how much data that u want for test the data here this is 0.2 means 20 percent of the data .

*stratify is Y actually Y basically as the values as either one and zero so we want our dataset to be splited into same proportion .

*if we dont include the startified then all the diabetes cases may go to x_train and all Non diabetic cases go to X_test.so that is the reason we use stratified to avoid that where there will be you know similar proportion of diabetic cases and non diabetic cases as they are in the original data set so that is the reason for stratifying .

```
[ ] print(X.shape,X_train.shape,X_test.shape)

    (768, 8) (614, 8) (154, 8)
```

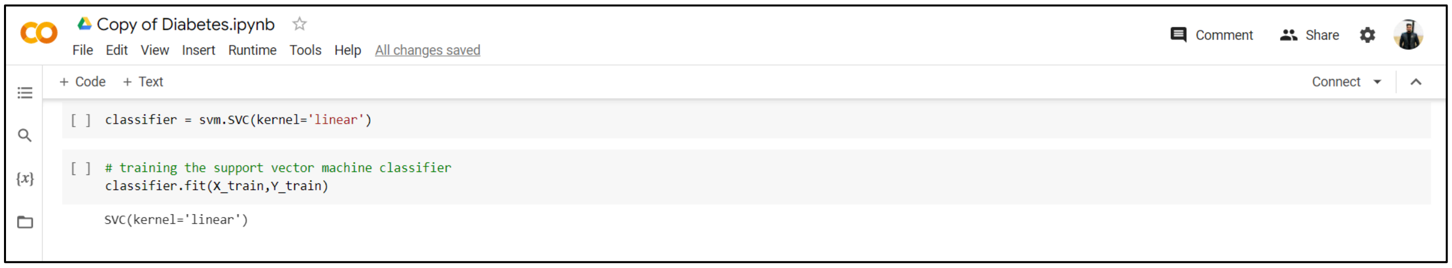
there r totally 768 examples in our original data set out of those 614 are going to be used for taining data and 154 are used for the test data . so this is a good proportion

Training the model

```
[ ] classifier = svm.SVC(kernel='linear')

[ ] # training the support vector machine classifier
```

Test size means how much data that u want for test the data here this is 0.2 means 20 percent of the data we test here .
stratify is Y actually Y basically as the values as either one and zero so we want our dataset to be splited into same proportion .

Apply Machine learning Algorithm

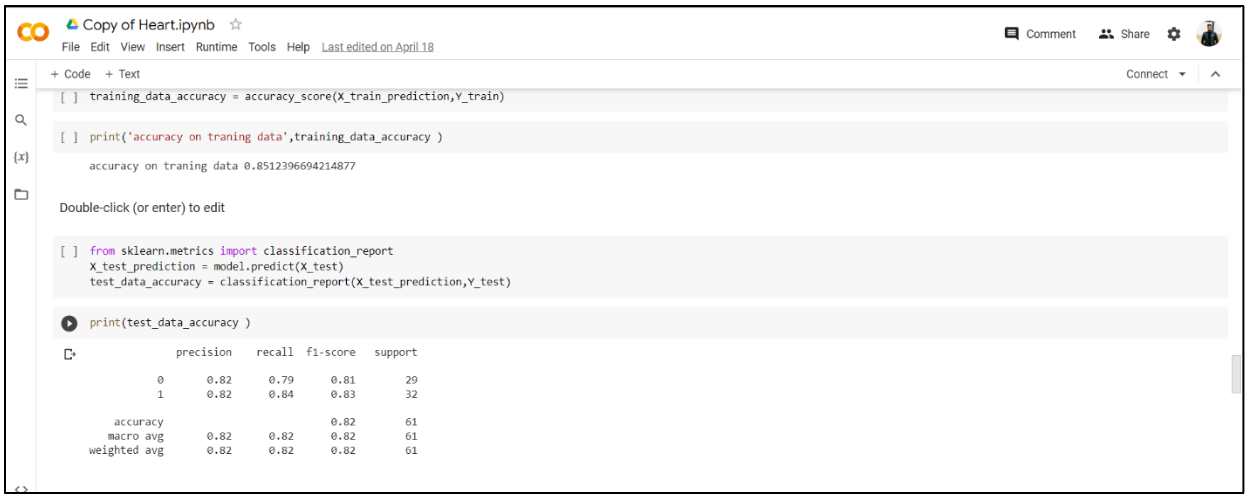


```
[ ] classifier = svm.SVC(kernel='linear')

[ ] # training the support vector machine classifier
    classifier.fit(X_train,Y_train)

    SVC(kernel='linear')
```

Model evaluation and Result

Accuracy without SMOTE

Accuracy is => 82 %


Copy of Heart.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 18
Comment Share
+ Code + Text
Connect

```
[ ] training_data_accuracy = accuracy_score(X_train_prediction,Y_train)

[ ] print('accuracy on traning data',training_data_accuracy )

    accuracy on traning data 0.8512396694214877
```

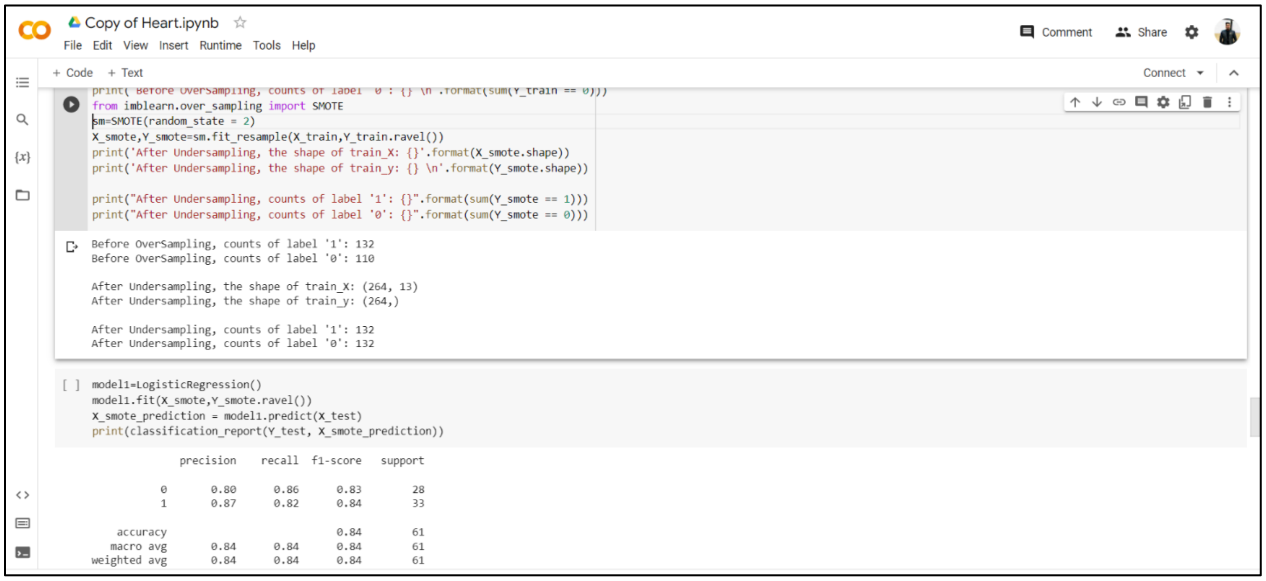
Double-click (or enter) to edit

```
[ ] from sklearn.metrics import classification_report
    X_test_prediction = model.predict(X_test)
    test_data_accuracy = classification_report(X_test_prediction,Y_test)

    print(test_data_accuracy )

              precision    recall  f1-score   support

           0       0.82      0.79      0.81        29
           1       0.82      0.84      0.83        32

    accuracy                           0.82        61
   macro avg       0.82      0.82      0.82        61
weighted avg       0.82      0.82      0.82        61
```

SMOTE :- Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in your dataset in a balanced way. The component works by generating new instances from existing minority cases that you supply as input.

Accuracy with SMOTE

Accuracy is => 84 %



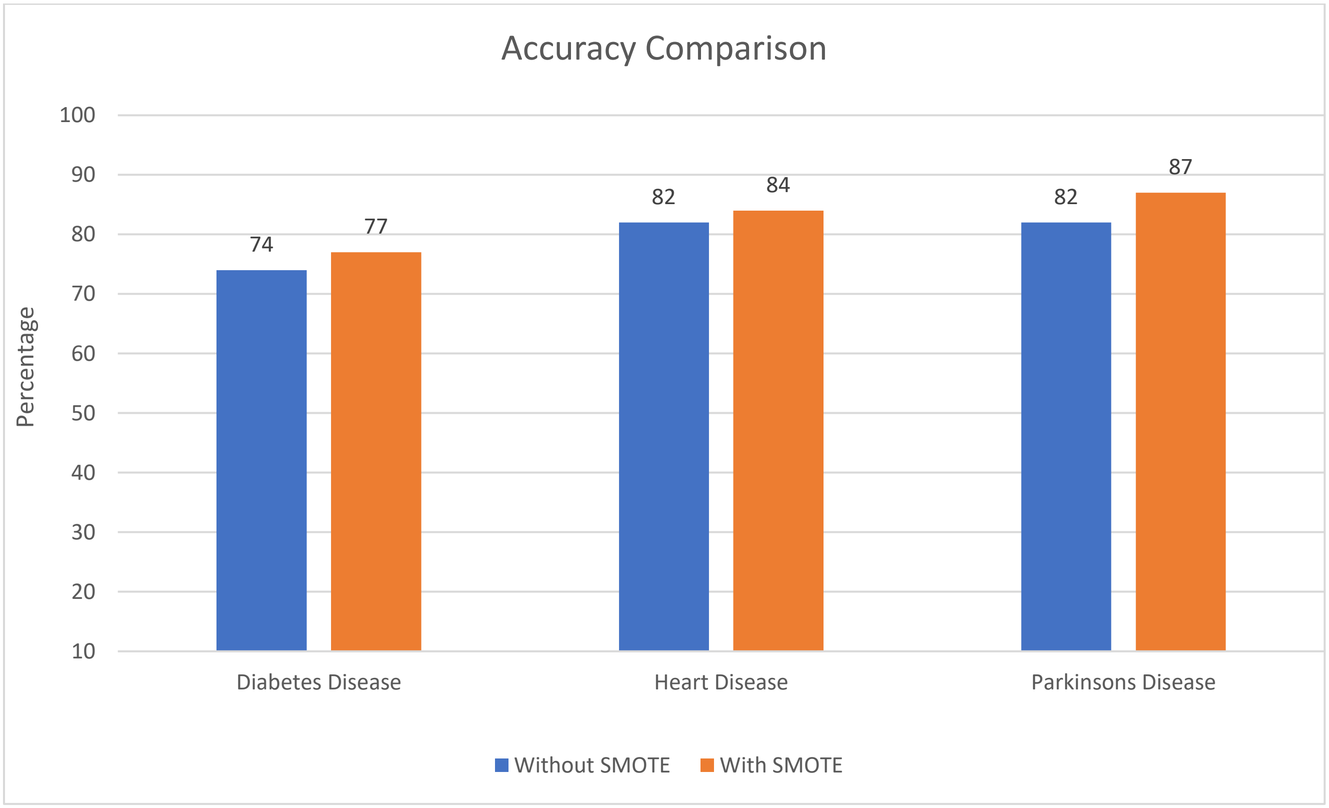
```
print('Before OverSampling, counts of label '0': {} \n'.format(sum(Y_train == 0)))
from imblearn.over_sampling import SMOTE
sm=SMOTE(random_state = 2)
X_smote,Y_smote=sm.fit_resample(X_train,Y_train.ravel())
print('After Undersampling, the shape of train_X: {}'.format(X_smote.shape))
print('After Undersampling, the shape of train_y: {} \n'.format(Y_smote.shape))

print("After Undersampling, counts of label '1': {}".format(sum(Y_smote == 1)))
print("After Undersampling, counts of label '0': {}".format(sum(Y_smote == 0)))
```

```
Before OverSampling, counts of label '1': 132
Before OverSampling, counts of label '0': 110

After Undersampling, the shape of train_X: (264, 13)
After Undersampling, the shape of train_y: (264,)

After Undersampling, counts of label '1': 132
After Undersampling, counts of label '0': 132
```

```
model1=LogisticRegression()
model1.fit(X_smote,Y_smote.ravel())
X_smote_prediction = model1.predict(X_test)
print(classification_report(Y_test, X_smote_prediction))
```

```
              precision    recall  f1-score   support

           0       0.80      0.86      0.83        28
           1       0.87      0.82      0.84        33

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61
```

Compare Accuracy with or without Smote



Conclusion

The main aim of this report is to predict the disease in accordance with parameters of Laboratory reports put down by the patients with proper implementation of Machine Learning algorithm. In this report we have used 2 Machine Learning algorithm for prediction and achieved the mean accuracy of more than 84% which shows remarkable rectification and high accuracy than previous work and also makes this system more reliable than the existing one for this job and hence provides better satisfaction to the user in comparison with the other one. It also stores the data entered by the user and the name of the disease the patient is suffering from in the Database which can be used as past record and will help in future for future treatment and thus contributing in easier health management .We have also created a GUI for better interaction with the system by users which is very easy to operate .This report shows that Machine Learning algorithm can be used to predict the disease easily with different parameters and models. In the end we can say that our system has no threshold of the users because everyone can use this system.

References :-

Datasets –

- Diabetes
- Heart
- Parkinson's

Theory-

- Machine Learning
- Supervised Learning

Software-

- Google Colab Notebook
- Anaconda.