General Disease Prediction System

Project report in partial fulfilment of the requirement for the award of the degree of

Master of Computer Applications

Submitted By

Manoranjan Panigrahi.
University Roll No. 304202000700004
Department of Computer Applications
UEM, Kolkata

Sayan Sen.
University Roll No. 304202000700037
Department of Computer Applications
UEM, Kolkata

Debapriya Mukherjee.
University Roll No. 304202000700055
Department of Computer Applications
UEM, Kolkata

Roshni Dey.
University Roll No. 304202000700008
Department of Computer Applications
UEM, Kolkata

Swarnali Ghosh.
University Roll No. 304202000700007
Department of Computer Applications
UEM, Kolkata

Under the guidance of

PROF. Kaustuv Bhattacharjee.
Department of Computer Applications
UEM, Kolkata

UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata – 700 156

CERTIFICATE

This is to certify that the project entitled **General Disease Prediction System** submitted by **Manoranjan Panigrahi (University Roll No. 304202000700004), Sayan Sen (University Roll No. 304202000700037)**, **Debapriya Mukherjee (University Roll No. 304202000700055), Roshni Dey (University Roll No. 304202000700008)** and **Swarnali Ghosh (University Roll No. 304202000700007)** Students of UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA, in fulfilment of requirement for the degree of Master of Computer Applications is a bona fide work carried out by them under the supervision and guidance of Prof. Kaustuv Bhattacharjee. during 3rd Semester of academic session of 2020-2022. The content of this report has not been submitted to any other university or institute for the award of any other degree.

I am glad to inform that the work is entirely original and its performance is found to be quite satisfactory.

Prof. Kaustuv Bhattacharjee.	Prof. Kaustuv Bhattacharjee
Assistant Professor	Head of the Department
Department of Computer Applications	Department of Computer Applications
UEM, Kolkata	UEM, Kolkata

ACKNOWLEDGEMENT

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide Prof. Kaustuv Bhattacharjee. of the Department of Computer Applications, UEM, Kolkata, for his wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

We would also like to express our sincere gratitude to Prof. Kaustuv Bhattacharjee, HOD, Department of Computer Applications, UEM, Kolkata and all other departmental faculties for their ever-present assistant and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Manoranjan Panigrahi.

Enrollment No. 12020007015004

Sayan Sen.

Enrollment No. 12020007015037

Debapriya Mukherjee.

Enrollment No. 12020007015055

Roshni Dey.

Enrollment No. 12020007015008

Swarnali Ghosh.

Enrollment No. 12020007015007

TABLE OF CONTENTS

ABSTRACT

This project aims to develop a web application for General Disease Prediction System. Our project aims to provide a user friendly platform to cross validate results at go and to spread general awareness and provide precautionary measures. With the advancement in technologies and mobile phones being the most used user-friendly device, our team has come with an application that provides a prediction of the three most caused lifestyle diseases like diabetes, heart disease and parkinson diseases at your hand. General Disease Prediction System (GDPS) allows you to make important predictions about the severity of an ongoing disease with few inputs given by user of parameters. In today's day time is the factor and being healthy is also an essential so this idea will help user and also encourage the idea of using Internet more widely.

CHAPTER – 1: INTRODUCTION

1.1 Project Title

General Disease Prediction System (GDPS).

1.2 Purpose

Our proposed General Disease Prediction System (GDPS) is for those who often feel reluctant to go to hospital or physician on minor symptoms. However, in many cases, these minor symptoms may trigger major health hazards. As online health advice is easily reachable, using GDPS can be a great head start for users. Moreover, existing online health care systems suffer from lack of reliability and accuracy. Herein, we propose a system that relies on guided user input. The system takes input from the user and provides a report of user's condition on mentioned diseases. Before doing anything we did a decent research on rapid escalation of Internet technology and data for which handheld devices has opened up new avenues for online healthcare system.

1.3 Objectives

The objectives of this study are summarized below:

1.3.1 General Objective

-To implement support vector machine or SVM and logistic regression that predicts the seriousness disease as per the input entered by the user.

1.3.2 Specific Objective

-The main objective of the project is to design and develop a user friendly, efficient and web application based General Disease Prediction System (GDPS).

-A reliable system to spread awareness among users.

-Computerization can be helpful as means of saving time.

CHAPTER – 2: LITERATURE SURVEY

Here we will elaborate the aspects like the literature survey of the project and what all projects are existing and been actually used in the market which the makers of this project took the inspiration from and thus decided to go ahead with the project covering with the problem statement.

Support Vector Machine:

A support vector machine (SVM) is machine learning algorithm that analyses data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs are used in text categorization, image classification, handwriting recognition and in the sciences.

Vladimir Vapnik said that-"First of all, SVM was developed over 30 years. The first publication we did jointly with Alexey Chervonenkis in 1963 and it was about optimal separating hyper-planes. It is actually linear SVM. In 1992, jointly with Bernhard Boser and Isabelle Guyon, we introduced the kernel trick and in 1995, jointly with Corinna Cortes, we introduced slack variables and it became SVM.I believe there are several reasons. First of all, it is effective. It gives very stable and good results. From a theoretical point of view, it is very clear, very simple, and allows many different generalizations, called kernel machines. It also introduced pattern recognition to optimization scientists and this includes new researchers in the field. There exist several very good libraries for SVM algorithms. All together these make SVM popular."

Logistic Regression:

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analysing the relationship between one or more existing independent variables.

Paul Allison said that – "For the analysis of binary data, logistic regression dominates all other methods in both the social and biomedical sciences. It wasn't always this way. In a 1934 article in *Science*, Charles Bliss proposed the probit function for analyzing binary data, and that method was later popularized in David Finney's 1947 book *Probit Analysis*. For many years, probit was the method of choice in biological research.

In a 1944 article in the *Journal of the American Statistical Association*, Joseph Berkson introduced the logit model (aka logistic regression model) and argued for its superiority to the probit model. The logistic method really began to take off with the publication of David Cox's 1970 book *Analysis of Binary Data*. Except for toxicology applications, probit has pretty much disappeared from the biomedical world. There are other options as well (like complementary log-log) but logistic regression is the overwhelming favorite."

CHAPTER – 3: PROBLEM STATEMENT & DISCUSSION

3.1 Problem Statement

It is estimated that more than 70% of people in India are prone to heart disease, diabetes and there is crude prevalence rate of 14.1 per 100,000 in Parkinson in every year. This may because many people don't realize that the general body diseases could be symptoms to something more harmful, 25% of the population succumbs to death because of ignoring the early general body symptoms. Hence identifying or predicting the disease at the earliest is very pivotal to avoid any unwanted casualties.

3.2 Discussion

The purpose of this system is to provide prediction for the general and more commonly occurring disease that when unchecked can turn into fatal disease. The system applies support vector machine or SVM and logistic regression algorithms. This system will predict the severity of mentioned diseases based on the given inputs and will show precautionary measures required to avoid the aggression of disease. It will also help the doctors to analyse the pattern of presence of diseases in the society. In this project, the disease prediction system will be trained using machine learning.

CHAPTER – 4: PROPOSED SOLUTION & RESULT ANALYSIS

4.1 PROPOSED SOLUTION

Concerning to the problem stated above we are going to implement support vector machine or SVM and logistic regression that predicts the seriousness of selected disease as per the input entered by the user. Besides this we aim to design and develop a user friendly and efficient web application based disease prediction System.

4.1.1 Feasibility Analysis

Technical feasibility

The project is technically feasible as it can be built using the existing available technologies. It is a web based applications that uses flask Framework. The technology required by GDPS is available and hence it is technically feasible.

Economic feasibility

The project is economically feasible as there is no cost involve in the project. As the data samples increases, our system will get more efficient.

Operational feasibility

The project is operationally feasible as the user having basic knowledge about computer and Internet. General Disease Prediction System is based on client-server architecture where client is users and server is the machine.

4.1.2 Requirement Analysis

Functional requirements

a. Predict disease with the given inputs by user.
b. Compare the given inputs with the input datasets and predicts the severity of disease.

Non-functional requirements

a. Display whether the user is affected by particular disease or not.

4.1.3 SYSTEM DESIGN

Methodology

Disease Prediction has been already implemented using different techniques like Neural Network, decision tree and various algorithms. So, our disease predictor uses SVM and logistic regression algorithms for the prediction of different diseases.

Data collection

Data collection has been done from the internet to identify the disease here the real symptoms of the disease are collected i.e. no dummy values are entered. The symptoms of the disease are collected from different health related websites.

Algorithm implemented

The algorithm implemented in this project is support vector machine or SVM and logistic regression algorithms.

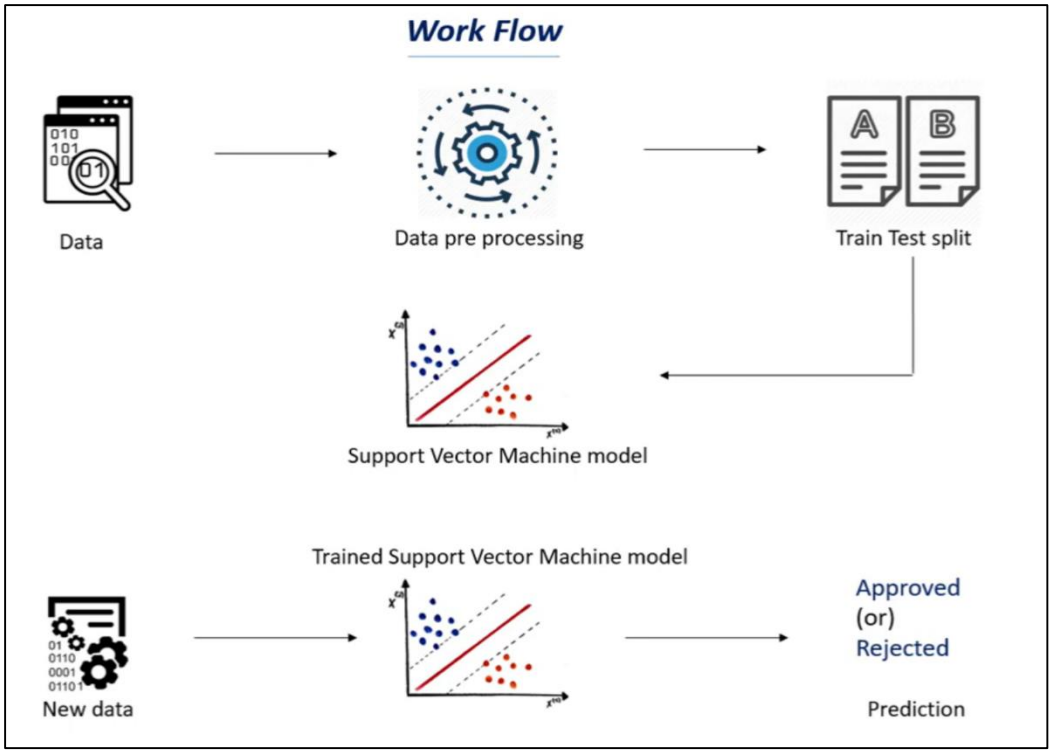
4.2 RESULT ANALYSIS

The system will initially be fed data from different sources, the data will then be pre-processed before further process is carried out, this is done to get clean data from the raw initial data, as the raw data would be noisy, or flawed. This data will be processed using algorithms, the system and will be trained to predict the disease based on the input data given by the user.

The work flow of Parkinson's disease is as follows:

Parkinson's disease a progressive nervous system disorder that affects movement leading to shaking, stiffness and difficulty with walking balance. It begins gradually and get worse overtime. It affects mostly male than female and more than 50-60 years of age.

We are building a system using machine learning algorithm SVM to diagnose this. We collected the Parkinson's data (about the patient who have Parkinson's and who doesn't have Parkinson's) which we need to train our machine learning model. Our machine learning model can understand the pattern of the data given by us about Parkinson's disease and understand what are the symptoms can be found in Parkinson's patient and what are the symptoms that are common in non-parkinson's patient. After having the dataset in hand we need to process this data as we cannot feed the raw data to our model. Hence we need to split our data into train and test model. We use training data in our machine learning model to train or make understand the model about the data. For this we are going to use SVM (Support Vector Machine Model). Here training data is used to train the model and testing data is used for evaluating our model. So after evaluation we will have a trained SVM model where by provided data it can detect whether a person has Parkinson or not. So, this is the workflow which we have followed.



Parkinson's Disease Detection:

Importing Dependencies:

These are the libraries or dependencies that we need.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
```

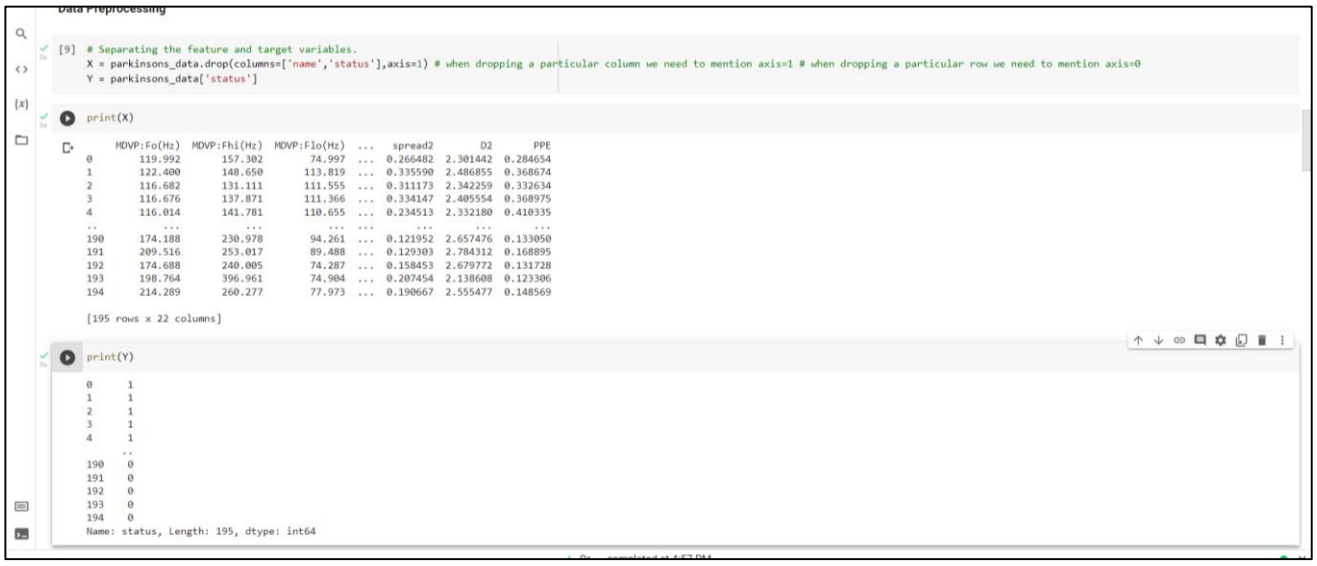
Data collection and analysis :

Here, we will load the data from csv file and analysed the data.

```
Data collection and analysis

[2] #loading the data from csv to panda dataframe
    parkinsons_data = pd.read_csv('/content/parkinsons.csv')

[3] # printing the first 5 rows of the dataframe
    parkinsons_data.head()

[4] #number of rows and coloumns in the dataframe.
    parkinsons_data.shape #shape function will give the number of rows and columnsin the dataset

    (195, 24)

[5] # getting more info about the dataset
    parkinsons_data.info()

    # checking for missing values in each column of the dataset
    parkinsons_data.isnull().sum()

[7] # getting some statistical measures about data
    parkinsons_data.describe()

    # distribution of target variable(means status)
    parkinsons_data['status'].value_counts()

[ ] # grouping the data based on the target variable
    parkinsons_data.groupby('status').mean()
```

Data Preprocessing :

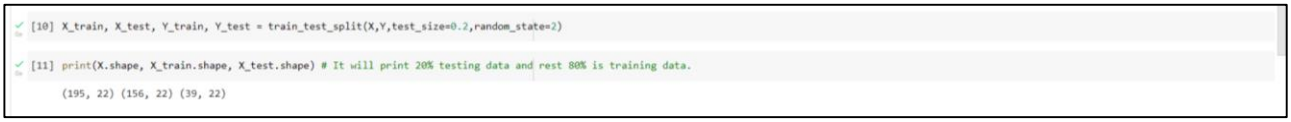
Data preprocessing in refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models.



```
Data Preprocessing

[9] # Separating the feature and target variables.
    X = parkinsons_data.drop(columns=['name','status'],axis=1) # when dropping a particular column we need to mention axis=1 # when dropping a particular row we need to mention axis=0
    Y = parkinsons_data['status']

print(X)

     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  ...   spread2        D2       PPE
0        119.992       157.302        74.997  ...  0.266482  2.301442  0.284654
1        122.400       148.650       113.819  ...  0.335590  2.486855  0.368674
2        116.682       131.111       111.555  ...  0.311173  2.342259  0.332634
3        116.676       137.871       111.366  ...  0.334147  2.405554  0.368975
4        116.014       141.781       110.655  ...  0.234513  2.332180  0.410335
..           ...           ...           ...  ...       ...       ...       ...
190      174.188       230.978        94.261  ...  0.121952  2.657476  0.133050
191      209.516       253.017        89.488  ...  0.129303  2.784312  0.168895
192      174.688       240.005        74.287  ...  0.158453  2.679772  0.131728
193      198.764       396.961        74.904  ...  0.207454  2.138608  0.123306
194      214.289       260.277        77.973  ...  0.190667  2.555477  0.148569

[195 rows x 22 columns]

print(Y)

0      1
1      1
2      1
3      1
4      1
      ..
190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```

Splitting the data into trainee data and Test data:

We are splitting the data into training data and testing data.



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

[11] print(X.shape, X_train.shape, X_test.shape) # It will print 20% testing data and rest 80% is training data.

     (195, 22) (156, 22) (39, 22)
```

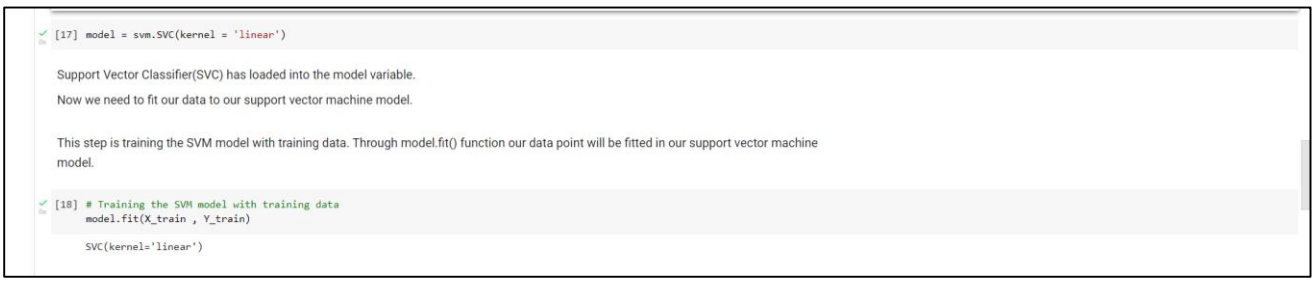
Data Standardization:

We need to stadarize the data in a normal form and to do that we use data stadarization technique.

```
[13] scaler.fit(X_train)

     StandardScaler()

[14] X_train = scaler.transform(X_train)

     X_test = scaler.transform(X_test)

print(X_train)
```

Model Training (SVM):

This step is training the SVM model with training data.



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

Support Vector Classifier(SVC) has loaded into the model variable.

Now we need to fit our data to our support vector machine model.

This step is training the SVM model with training data. Through model.fit() function our data point will be fitted in our support vector machine model.

[18] # Training the SVM model with training data
     model.fit(X_train , Y_train)

     SVC(kernel='linear')
```

Model Evaluation:

This is about model evaluation where accuracy score of model will be found.

```
[19] # Accuracy Score on training data
     X_train_prediction = model.predict(X_train)  # As the model is trained, Now we can give the features and try to predict whether a person has Parkinson's or not. It will give the value as either 1 or 0.
     training_data_accuracy = accuracy_score(Y_train, X_train_prediction) # Now we are compairing Y_train(Original Value) with X_train(Value predicted by our model) to find our training data accuracy.
                                                                          # Accuracy function is imported from this sklearn.matrix.
```



```
[20] print('Accuracy of training data : ', training_data_accuracy)
```

Prediction:

Here, the prediction is printed in the form of a list. The name of the list is prediction.

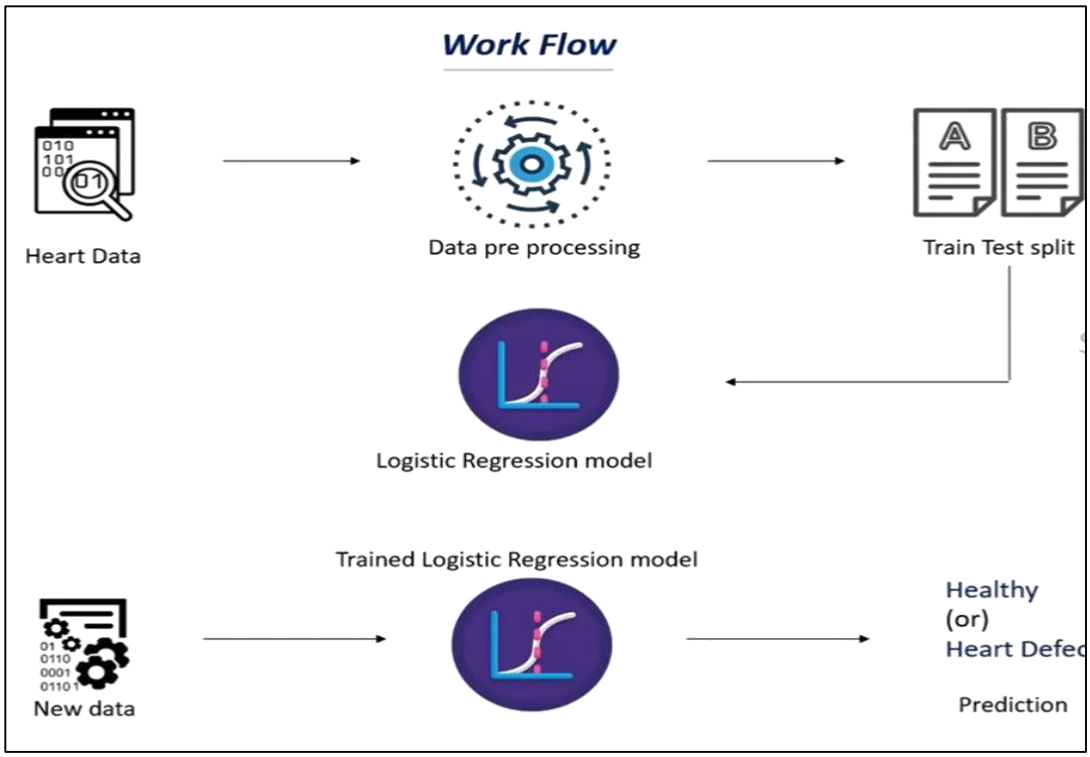
```
input_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.422229,0.741367,-7.348300,0.177551,1.743867,0.085569)

# Changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

#reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the data
std_data = scaler.transform(input_data_reshaped)

prediction = model.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print("The Person does not have Parkinsons Disease")

else:
  print("The Person has Parkinsons")
```

```
[0]
The Person does not have Parkinsons Disease
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  "X does not have valid feature names, but"
```

The work flow of Heart disease is as follows:

The term "heart disease" refers to several types of heart conditions. The most common type of heart disease in the United States is coronary artery disease (CAD), which affects the blood flow to the heart. Decreased blood flow can cause a heart attack. Your risk for heart disease increases with age, especially with people of colour and for those who are over 65. While the average age for a heart attack is 64.5 for men, and 70.3 for women, nearly 20 percent of those who die of heart disease are under the age of 65.

To diagnose this we are developing a system using machine learning algorithm which is Logistic Regression. At first we collect the heart data (about the patient who have Heart disease and who doesn't have Heart disease). This dataset contains several health parameters which correspond to a person's healthiness of the heart. Once we have this dataset we need to process this dataset because we can't feed this raw data into our machine learning algorithm. We need to process this dataset to make it fit & compatible for our machine learning algorithm to learn. Once we process the data we need to split our data into training data & testing data. This is because we often train our machine learning algorithm with training data & we will evaluate our model. We will evaluate the performance of our model using the test data, this part is called a train test split where we will split our original dataset into training & test data. Once we do that we will feed our training data to our machine learning model. In this case we are going to use logistic regression model because this particular use case is a binary classification. We are going to classify whether a person has a heart disease or not. This is a binary classification either yes or no kind of questions & in that binary classifications logistic regression model is very useful. It's the best model when it comes to binary classification once we train this logistic regression model with our training data. We will do some evaluation on our model to check its performance. After that we will get a trained logistic regression model & to this model when we feed new data our model can predict whether that person has heart disease or not. So, this is the workflow which we have followed.



Heart Disease Detection:

Importing Dependencies :

These are the libraries or dependencies that we need.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data collection & Processing:

Here, we will load the data from csv file and analysed the data.

```
# loading the csv data to a Pandas DataFrame
heart_data = pd.read_csv('/content/data.csv')
```

```
# print first 5 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 5 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
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

```
# number of rows and columns in the dataset
heart_data.shape
```

```
(303, 14)
```

```
# getting some info about the data
heart_data.info()
```

```
<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  
dtypes: float64(1), int64(13)
```

```
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
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

```
# checking the distribution of Target Variable
heart_data['target'].value_counts()
```

```
1    165
0    138
Name: target, dtype: int64
```

1 --> Defective Heart

0 --> Healthy Heart

Splitting the Features and Target:

We are splitting the data into feature and target.

```
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

```
print(X)
```

```
     age  sex  cp  trestbps  chol  ...  exang  oldpeak  slope  ca  thal
0     63    1   3       145   233  ...      0      2.3      0   0     1
1     37    1   2       130   250  ...      0      3.5      0   0     2
2     41    0   1       130   204  ...      0      1.4      2   0     2
3     56    1   1       120   236  ...      0      0.8      2   0     2
4     57    0   0       120   354  ...      1      0.6      2   0     2
..   ...  ...  ..       ...   ...  ...    ...      ...    ...  ..   ...
298   57    0   0       140   241  ...      1      0.2      1   0     3
299   45    1   3       110   264  ...      0      1.2      1   0     3
300   68    1   0       144   193  ...      0      3.4      1   2     3
301   57    1   0       130   131  ...      1      1.2      1   1     3
302   57    0   1       130   236  ...      0      0.0      1   1     2

[303 rows x 13 columns]
```

```
print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Splitting the data into training data and Test data:

We are splitting the data into training data and testing data.

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

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

```
(303, 13) (242, 13) (61, 13)
```

Model Training (Logistic Regression):

This step is training the SVM model with training data.

```
model = LogisticRegression()
```

```
# training the LogisticRegression model with Training data
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Model Evaluation:

This is about model evaluation where accuracy score of model will be found.

```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy on Training data : ', training_data_accuracy)
```

Prediction:

Here, the prediction is printed in the form of a list. The name of the list is prediction.

```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

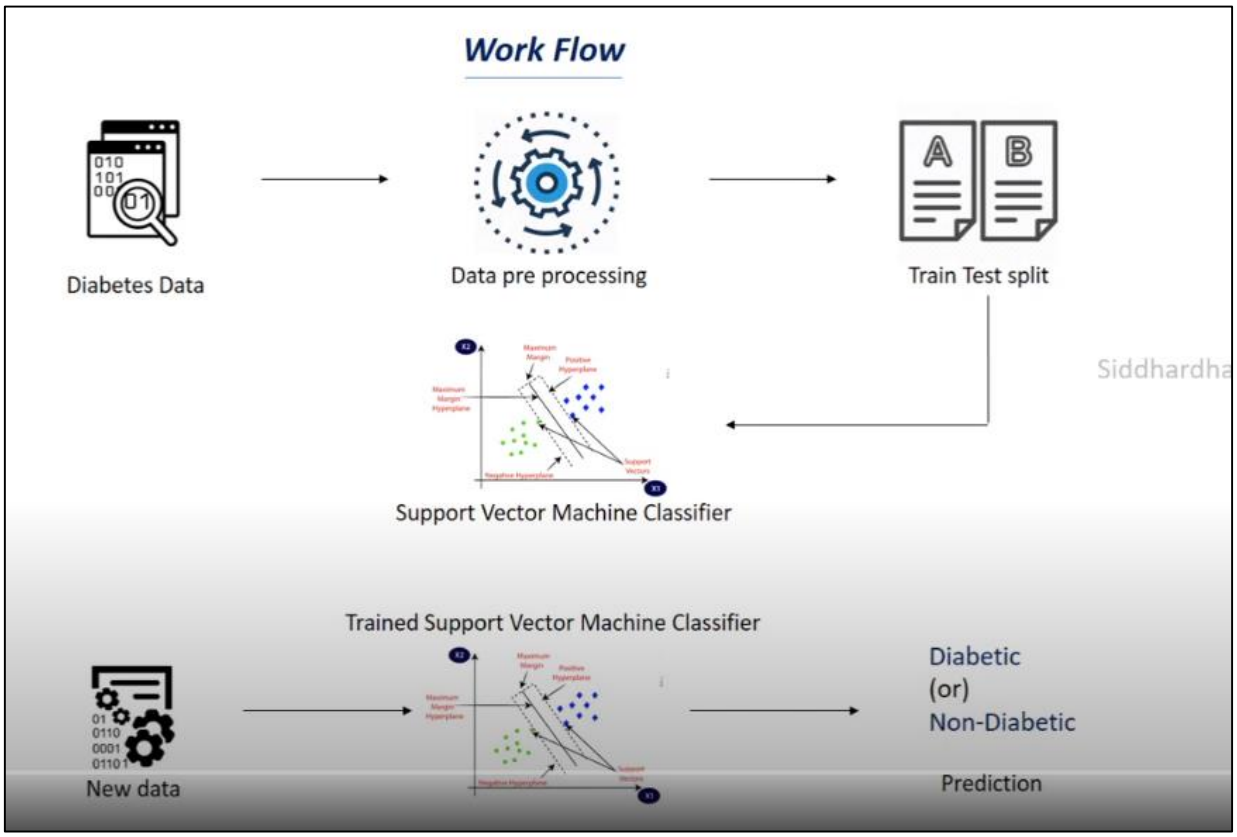
```
[0]
The Person does not have a Heart Disease
```

The work flow of Diabetes disease is as follows:

Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Insulin is a hormone that regulates blood sugar. Hyperglycemia, or raised blood sugar, is a common effect of uncontrolled diabetes and over time leads to serious damage to many of the body's systems, especially the nerves and blood vessels. IN 2014, 8.5% of adults aged 18 years and older had diabetes. In 2019, diabetes was the direct cause of 1.5 million deaths. To present a more accurate picture of the deaths causes by diabetes, however, deaths due to higher-than-optimal blood glucose through cardiovascular disease, chronic kidney disease and tuberculosis should be added. In 2012 (year of the latest available data), there were another 2.2 million deaths due to high blood glucose.

For this disease we train our model with several medical information such as the blood glucose level and insulin level of patients along with whether the person has diabetic or non-diabetic.

Once we feed the data to our vector machine model what happens is it tries to plot the data in a graph and once it plots the data it tries to find a hyper plane. This hyperplane separates these two data, so once we feed new data in the model it tries to put that particular data in either of these two groups : Diabetic or Non-diabetic.



Diabetes Disease Detection:

Importing Dependencies :

These are the libraries or dependencies that we need.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data collection & Processing:

Here, we will load the data from csv file and analysed the data.

```
# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('/content/diabetes.csv')
```

```
pd.read_csv?
```

```
# printing the first 5 rows of the 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

```
# number of rows and Columns in this dataset
diabetes_dataset.shape
```

```
(768, 9)
```

```
# getting the statistical measures of the data
diabetes_dataset.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	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	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	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

```
diabetes_dataset['Outcome'].value_counts()
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

0 --> Non-Diabetic

1 --> Diabetic

```
diabetes_dataset.groupby('Outcome').mean()
```

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

```
# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

```
print(X)
```

```
     Pregnancies  Glucose  BloodPressure  ...   BMI  DiabetesPedigreeFunction  Age
0              6      148             72  ...  33.6                     0.627   50
1              1       85             66  ...  26.6                     0.351   31
2              8      183             64  ...  23.3                     0.672   32
3              1       89             66  ...  28.1                     0.167   21
4              0      137             40  ...  43.1                     2.288   33
..           ...      ...            ...  ...   ...                       ...  ...
763           10      101             76  ...  32.9                     0.171   63
764            2      122             70  ...  36.8                     0.340   27
765            5      121             72  ...  26.2                     0.245   30
766            1      126             60  ...  30.1                     0.349   47
767            1       93             70  ...  30.4                     0.315   23

[768 rows x 8 columns]
```

```
print(Y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Data Standardization:

We need to standardize the data in a normal form and to do that we use data stadarization technique.

```
scaler = StandardScaler()
```

```
scaler.fit(X)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

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

```
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]]
```

```
X = standardized_data
Y = diabetes_dataset['Outcome']
```

```
print(X)
print(Y)
```

```
[[ 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]]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Splitting the data into training data and Test data:

We are splitting the data into training data and testing data.

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

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

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

Model Training (SVM):

This step is training the SVM model with training data.

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

```
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Model Evaluation:

This is about model evaluation where accuracy score of model will be found.

```
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

Prediction:

Here, the prediction is printed in the form of a list. The name of the list is prediction.

```
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
[[ 0.3429808   1.41167241  0.14964075 -0.09637905  0.82661621 -0.78595734
   0.34768723  1.51108316]]
[1]
The person is diabetic
```

CHAPTER – 5: CONCLUSION & FUTURE SCOPE

Conclusion

This project aims to predict the disease on the basis of the inputs given by the user. The project is designed in such a way that the system takes input from the user and provides a report of user's condition i.e. predict disease's prevalence in body. Average prediction accuracy probability of heart disease is 77%, diabetes is 81% and Parkinson is 88%. Disease Predictor was successfully implemented using flask framework.

Future Scope

i) As the current system covers only three diseases the plan is to include disease of higher fatality, like various cancers, so that early prediction and treatment could be done.

ii) This project has not implemented recommendation of medications to the user. So, medication recommendation can be implemented in the project.

iii) History about the disease for a user can be kept as a log and recommendation can be implemented for medications.

CHAPTER – 6: BIBLIOGRAPHY

1. https://www.neurologyindia.com/article.asp?issn=0028-3886;year=2018;volume=66;issue=7;spage=26;epage=35;aulast=Radhakrishnan
2. https://www.thelancet.com/journals/langlo/article/PIIS2214-109X(21)00214-X/fulltext
3. https://www.frontiersin.org/articles/10.3389/fneur.2020.00524/full
4. https://www.researchgate.net/publication/288324436_Global_Prevalence_and_Therapeutic_Strategies_for_Parkinson's_Disease
5. https://www.irjet.net/archives/V5/i3/IRJET-V5I3930.pdf
6. https://www.kaggle.com/nidaguler/parkinsons-data-set
7. https://colab.research.google.com/drive/1PTbvUuqlOoCIr4uH0wsuVAsr88dtjb1c#scrollTo=Ojx1z6YnIKlT&uniqifier=2
8. https://www.mayoclinic.org/diseases-conditions/diabetes/diagnosis-treatment/drc-20371451#:~:text=A%20fasting%20blood%20sugar%20level,separate%20tests%2C%20you%20have%20diabetes
9. https://www.who.int/news-room/fact-sheets/detail/diabetes
10. https://www.healthline.com/health/diabetes
11. https://www.dropbox.com/s/uh7o7uyeghqkhoy/diabetes.csv?dl=0
12. https://colab.research.google.com/drive/1oxnhMTlomJ4HVhPuowpPFyMt1mwuOuQo?usp=sharing#scrollTo=-71UtHzNVWjB
13. https://colab.research.google.com/drive/1FYGPRSEGvd0urNlZmRJHx-gq6ANn3IpX?usp=sharing
14. https://statisticalhorizons.com/whats-so-special-about-logit
15. https://searchbusinessanalytics.techtarget.com/definition/logistic-regression
16. https://www.learningtheory.org/learning-has-just-started-an-interview-with-prof-vladimir-vapnik/