



# **“Predictive Modelling of Diabetes in Females”**

---

**Cloud - AI**

## **Author:**

Sujin S R (UB:23037526) ([ssreeku4@bradford.ac.uk](mailto:ssreeku4@bradford.ac.uk))

# **Abstract**

The aim of this project is to create a predictive model that uses a support vector machine classifier algorithm from machine learning library scikit-learn to check whether the person is diabetic or not. For that we use real-world data of women from kaggle, such as their pregnancy count, glucose tolerance, and other health-related characteristics. The model will be able to identify an individual's risk of developing diabetes using this data. Early detection is the aim to create awareness among people to start taking care of their health earlier. It's about using AI technology to keep people healthy by predicting their health and avoiding problems before they get serious.

## Table of Contents:

|                                       |    |
|---------------------------------------|----|
| Introduction.....                     | 4  |
| Problem Description.....              | 5  |
| Project Motivation and Objective..... | 7  |
| Dataset Description.....              | 8  |
| Workflow.....                         | 8  |
| Dataset Preprocessing.....            | 9  |
| Training the Model.....               | 14 |
| Predictive Analytics.....             | 19 |
| Conclusion.....                       | 21 |
| References.....                       | 22 |

# Introduction:

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 glucose. Hyperglycaemia, also called raised blood glucose 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 and 48% of all deaths due to diabetes occurred before the age of 70 years. Another 460 000 kidney disease deaths were caused by diabetes, and raised blood glucose causes around 20% of cardiovascular deaths.

Between 2000 and 2019, there was a 3% increase in age-standardised mortality rates from diabetes. In lower-middle-income countries, the mortality rate due to diabetes increased 13%.

By contrast, the probability of dying from any one of the four main noncommunicable diseases (cardiovascular diseases, cancer, chronic respiratory diseases or diabetes) between the ages of 30 and 70 decreased by 22% globally between 2000 and 2019.

In this project, we're using the machine learning model to help us figure out if someone has diabetes or not, by feeding the model such health related data of the person and it predicts the output. Using real data of 768 women from kaggle, the dataset includes counts of pregnancy, their glucose levels and other health related characteristics. Our goal is to identify these health issues so people can take care of their health. By doing this, people can find out if they have diabetes, by this information they can head start to do things to stay healthy. The Support Vector Machine Classifier algorithm from scikit-learn library is used for creating our model, this algorithm performs well on working with two-group classification problems.

## **Problem Description:**

Diabetes is a health issue that makes the body confused, especially how they handle sugar. Having too much sugar in the body makes the person feel tired, thirsty and it messes with the blood sugar. Also it can cause serious problems like it can make heart and blood vessels sick, that leads to heart attacks or strokes. So it's important for people to know about it at an early stage so that they can stay healthy. The solution is to develop a machine learning model that can predict whether a person has diabetes or not with the help of various health related parameters.

These are the long-term problems that can develop gradually, and can lead to serious damage if they go unchecked and untreated.

### **Eye Problems(retinopathy):**

Some people with diabetes develop an eye disease called diabetic retinopathy which can affect their eyesight. If retinopathy is picked up – usually from an eye screening test - it can be treated and sight loss prevented.

### **Foot Problems:**

Diabetes foot problems are serious and can lead to amputation if untreated. Nerve damage can affect the feeling in your feet and raised blood sugar can damage the circulation, making it slower for sores and cuts to heal. That's why it's important to tell your GP if you notice any change in how your feet look or feel.

### **Heart attack and stroke:**

When you have diabetes, high blood sugar for a period of time can damage your blood vessels. This can sometimes lead to heart attacks and strokes.

### **Kidney Problems(nephropathy):**

Diabetes can cause damage to your kidneys over a long period of time making it harder to clear extra fluid and waste from your body. This is caused by high blood sugar levels and high blood pressure. It is known as diabetic nephropathy or kidney disease.

**Nerve Damage(neuropathy):**

Some people with diabetes may develop nerve damage caused by complications of high blood sugar levels. This can make it harder for the nerves to carry messages between the brain and every part of our body so it can affect how we see, hear, feel and move.

**Gum Disease and other mouth problems:**

Too much sugar in your blood can lead to more sugar in your saliva. This brings bacteria which produces acid which attacks your tooth enamel and damages your gums. The blood vessels in your gums can also become damaged, making gums more likely to get infected.

**Related conditions like cancer:**

If you have diabetes, you're more at risk of developing certain cancers. And some cancer treatments can affect your diabetes and make it harder to control your blood sugar.

**Sexual Problems in Women:**

Damage to blood vessels and nerves can restrict the amount of blood flowing to your sexual organs so you can lose some sensation. If you have high blood sugar, you are also more likely to get thrush or a urinary tract infection.

**Sexual Problems in Men:**

The amount of blood flowing to your sexual organs can be restricted which may cause you to have difficulty getting aroused. It may lead to erectile dysfunction, sometimes called impotence.

## **Project Motivation and Objective:**

The ongoing evolution in medicine and, in particular, in the field of diabetology is strongly intertwined to a series of changes and innovations. The term “digital health” is a “container,” grouping together informatics and telecommunications that have the common objectives of diagnosis, treatment, or monitoring of diseases, maintenance of health and well-being, and support for healthy lifestyles. The US Food and Drug Administration has compiled a list of software programs with medical device functions (apps for smartphones and personal computers with diagnostic, monitoring, or therapeutic objectives), advanced business intelligence data analysis tools, artificial intelligence (AI), cloud, cyber security, and innovative health technologies. These tools belong to an evolving reality; they are still unclear but they are potentially leading to challenging and promising scenarios.

The scope of this position statement is to analyse the most relevant aspects and describe the changes that have already occurred or those that will take place shortly by exploring the possibilities of application and development in the field of diabetology. Indeed, the digital world is constantly expanding and it has already become an integral part of our personal and professional life. Smartphones, personal computers, and network access are now essential tools for almost all populations and for many diabetologists, both as individuals and professionals.

The main objective is to develop an AI model that can predict whether the individual is diabetic or not by feeding the model with various health parameters of the individual.

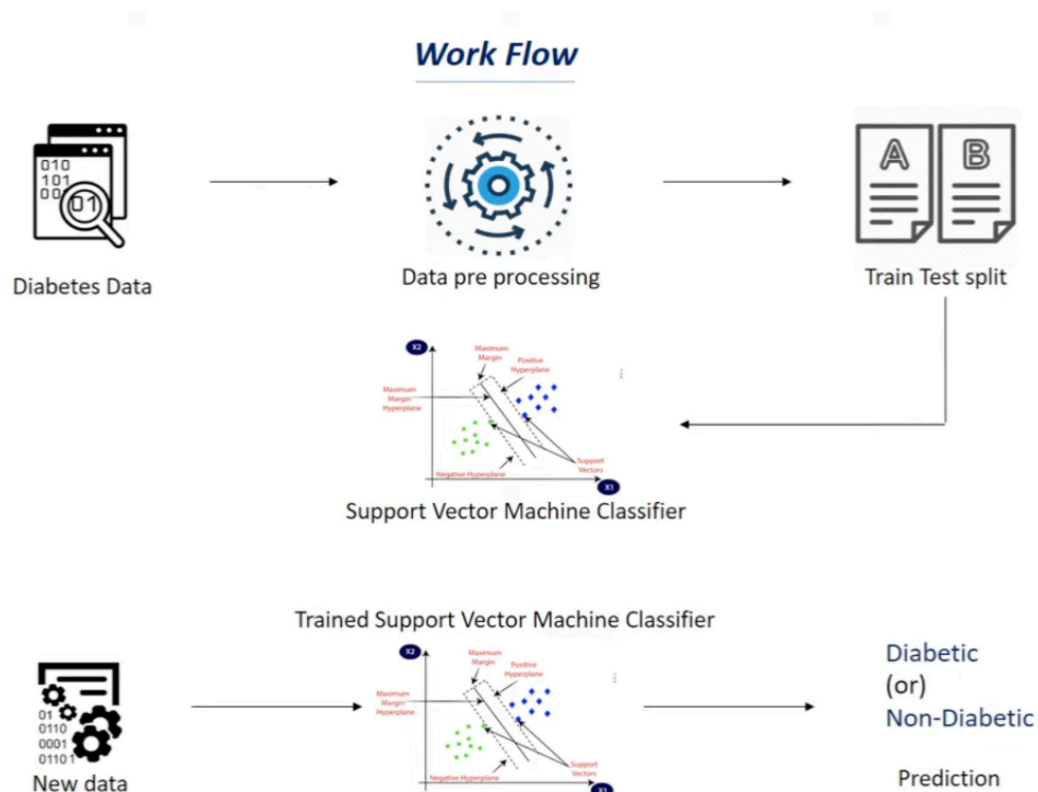
# Dataset Description:

The dataset is from [kaggle](https://www.kaggle.com). It consists of several medical independent variables that have 9 columns and 768 rows of real life data of women about their Pregnancies(number of times pregnant), Glucose(plasma glucose concentration level), Blood Pressure level, Skin Thickness(triceps skin fold thickness), Insulin level(2 hour serum insulin level), BMI(body mass weight), Diabetes Pedigree Function, Age and Outcome(diabetes-1 or non-diabetes-0).

## Workflow:

### Libraries used:

- Numpy
- Pandas
- Sklearn





First loading the data into the google colab by using pandas library then pre-process the data to many statistical measures that can be helpful for building models. Then splitting the data into two variables, one is for training dataset and test dataset, first step includes feeding the training data into our model using a support vector machine classifier algorithm then we train it and evaluate the accuracy of the model. After that, test data will be feeded to the model to test whether it predicts the new unknown test data correctly or not. Also by checking the accuracy of the model using test data we can evaluate if the model is fine tuned correctly or not. It involves multiple steps through these workflow will define everything briefly in the report step by step

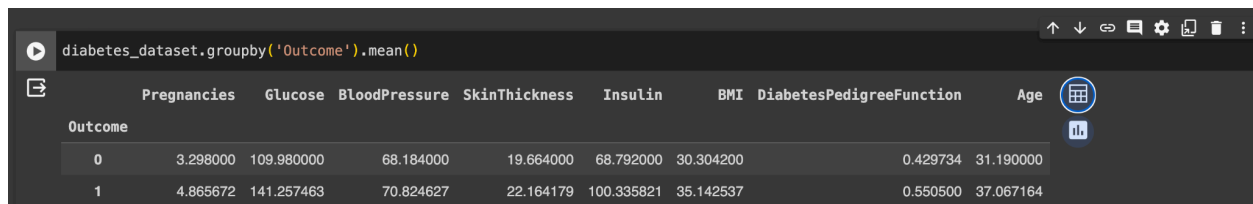
## Dataset Preprocessing:

Using the pandas first we load the database into the google colab, then analyse the data's statistical measures. By `dataset.describe()` function that is widely used in the pandas library to analyse central tendency and dispersion of the dataset. Also using this we can analyse the mean of the each column values, count of the total number of non-empty values in the column values, standard deviation of the column values, minimum value from the column and the maximum value from the column values, also the percentile of each column values.

```
# statistical measures
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   |

And let's do the main calculation for our model to understand how it can predict from the data for that, the column outcome shows the person is diabetic or not so if taking mean value of diabetic and non-diabetic we analyse from that what differences are there



The screenshot shows a Jupyter Notebook interface with a code cell containing the command `diabetes_dataset.groupby('Outcome').mean()`. Below the code cell, a table displays the mean values for various features, grouped by the 'Outcome' column (0 for non-diabetic, 1 for diabetic).

|         | Pregnancies | Glucose    | BloodPressure | SkinThickness | Insulin    | BMI       | DiabetesPedigreeFunction | Age       |
|---------|-------------|------------|---------------|---------------|------------|-----------|--------------------------|-----------|
| Outcome |             |            |               |               |            |           |                          |           |
| 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 |

From this mean value of data, the glucose level of each individual that has diabetes is greater than the one who doesn't have diabetes, also the insulin level for someone who has diabetes is greater than someone who doesn't have diabetes. Also, the age variation between the diabetes and non-diabetics individuals. And this calculation plays an important role for our model to predict from the data that the individual has diabetes or not.

Now the next step is to separate the data into two variables X and Y. X which holds the whole dataset except the 'Outcome' column and Y holds the Outcome column data only.

```
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

After separating these data into two variables, The main part of the data pre-processing is data standardisation is an important technique that is mostly performed as a pre-processing step before inputting data into many machine learning models, to standardise the range of features of an input data set.

Which means, down-scaling the values to a scale common to all, usually in the range -1 to 1 and keeping the range between the values intact. The mathematical formula for this step is  $Z\text{-Score} = (\text{Current\_value} - \text{Mean}) / \text{Standard Deviation}$ .

$$z = \frac{x_i - \mu}{\sigma}$$

Using this formula we are replacing all the input values by the Z-score for each and every value. Then the result would be values ranging from -1 to 1.

By using the scikit-learn library we have an inbuilt function for standardisation which is `StandardScaler()` and it is imported from the `sklearn.preprocessing` module. Storing this function in a variable and then invoke a `fit()` which is an inbuilt function and pass our `X` variable that has the separated data in it. Then store the data in a variable called `standardized_data` and use the `transform()` function passing the `X` variable. The result would be standardised data.

```
[ ] 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]]
```

When we build machine learning models in python, the Scikit Learn package gives us tools to perform common machine learning operations. One such tool is

the `train_test_split` function. The Sklearn `train_test_split` function helps us create our training data and test data. This is because typically, the training data and test data come from the same original dataset.

To get the data to build a model, we start with a single dataset, and then we split it into two datasets: train and test. So the scikit learn train test split function enables us to split a dataset into training data and test data. Having said that, how the function works depends on what data you give it as an input, and how to use the optional parameters in the syntax.

Next step is splitting the training data and the test data into four variables that are `X_train`, `X_test`, `Y_train`, `Y_test`. Using the `train_test_split()` is a function from the scikit-learn library used for splitting datasets into two parts: one for training a machine learning model and the other for testing the model's performance.

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

The train/test split function has a few important inputs and parameters

- `X`
- `Y`
- `test_size`
- `random_state`
- `stratify`

Let's look at each of these.

### **X (REQUIRED)**

The `X` argument is an input array that contains the feature data (i.e., the variables/columns you want to use to build your model. This is typically a Numpy array, but the function will allow other structures like Python lists. This object should be 2-dimensional, so if it's a 1-dimensional Numpy array, you may need to reshape your data. This is commonly done with the code `.reshape(-1,1)`.

## **Y**

The `y` argument typically contains the vector of target values (i.e., the target or label of your data). This is typically a 1-dimensional Numpy array, although the function will allow 2D arrays and lists. If it's a 1-dimensional object, it should have the same length as the number of rows in `X`. Or if it's 2D, it should have the same number of rows as `X`.

## **ADDITIONAL ARRAYS**

Although we commonly use `train_test_split` on one or two input arrays (`X` and `y`), technically, the function will operate on multiple arrays.

If you need to split many arrays, you can provide them in addition to `X` and `y`.

## **TEST\_SIZE**

The `test_size` parameter enables you to specify the size of the output test set. As an argument to this parameter, you can provide either an integer, or a float. If the argument is an integer, the size of the test set will be equal to that number. If the argument is a float, it must be between 0 and 1, and the number will represent the proportion of observations that will be in the test set. (Note: this is probably the most common way to use the parameter.)

## **RANDOM\_STATE**

The `random_state` parameter controls how the pseudo-random number generator randomly selects observations to go into the training set or test set. If you provide an integer as the argument to this parameter, then `train_test_split` will shuffle the data in the same order prior to the split, every time you use the function with that same integer. Effectively, if you provide an integer to this parameter, it will make your code exactly reproducible across every call of the function.

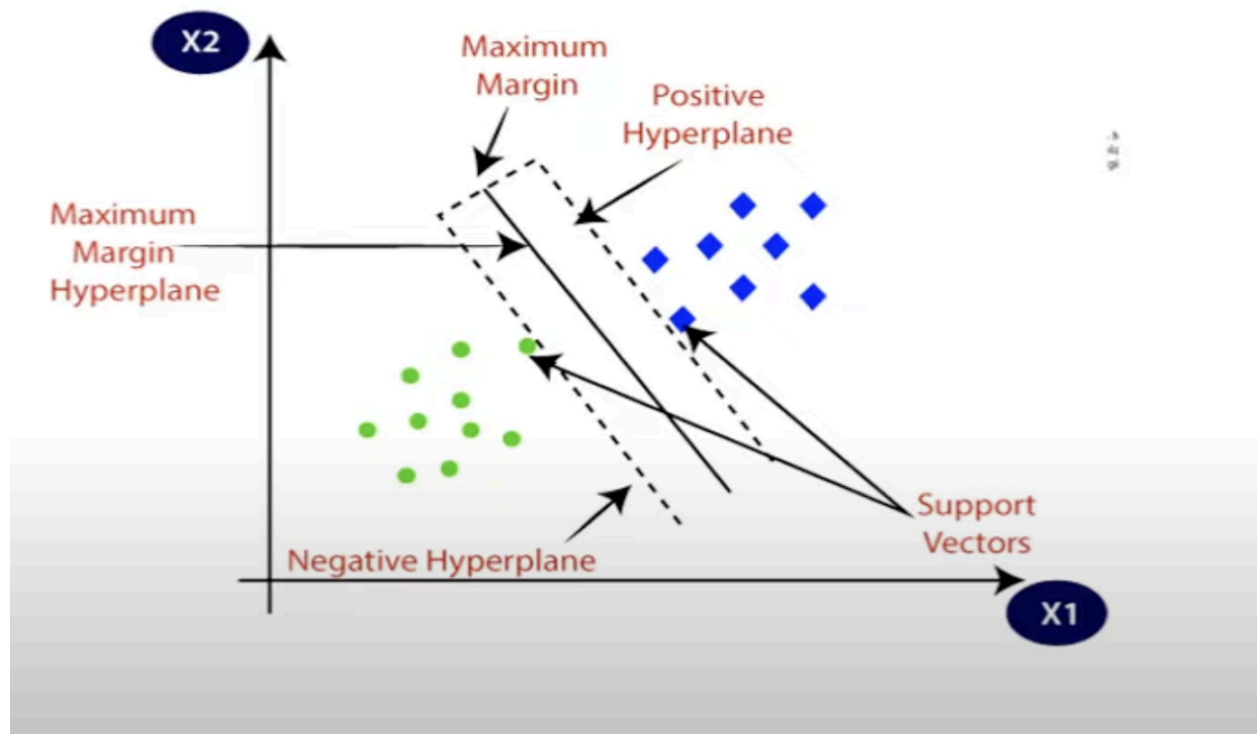
## **STRATIFY**

The `shuffle` parameter controls if the data are split in a stratified fashion. By default, this is set to `stratify = None`.

## Training the Model:

For training the model, Support Vector Machine classifier algorithm which is from the scikit learn library that uses supervised learning models to solve complex classification, regression, and outlier detection problems by performing optimal data transformations that determine boundaries between data points based on predefined classes, labels, or outputs. SVMs are widely adopted across disciplines such as healthcare, natural language processing, signal processing applications, and speech & image recognition fields.

### *Support Vector Machine*



Technically, the primary objective of the SVM algorithm is to identify a hyperplane that distinguishably segregates the data points of different classes. The hyperplane is localised in such a manner that the largest margin separates the classes under consideration.

As seen in the above figure, the margin refers to the maximum width of the slice that runs parallel to the hyperplane without any internal support vectors. Such hyperplanes are easier to define for linearly separable problems; however, for real-life problems or scenarios, the SVM algorithm tries to maximise the margin between the support vectors, thereby giving rise to incorrect classifications for smaller sections of data points.

SVMs are potentially designed for binary classification problems. However, with the rise in computationally intensive multiclass problems, several binary classifiers are constructed and combined to formulate SVMs that can implement such multiclass classifications through binary means.

In the mathematical context, an SVM refers to a set of ML algorithms that use kernel methods to transform data features by employing kernel functions. Kernel functions rely on the process of mapping complex datasets to higher dimensions in a manner that makes data point separation easier. The function simplifies the data boundaries for non-linear problems by adding higher dimensions to map complex data points.

While introducing additional dimensions, the data is not entirely transformed as it can act as a computationally taxing process. This technique is usually referred to as the kernel trick, wherein data transformation into higher dimensions is achieved efficiently and inexpensively.

The idea behind the SVM algorithm was first captured in 1963 by Vladimir N. Vapnik and Alexey Ya. Chervonenkis. Since then, SVMs have gained enough popularity as they have continued to have wide-scale implications across several areas, including the protein sorting process, text categorization, facial recognition, autonomous cars, robotic systems, and so on.

Kernel support vector machines (SVMs) are a variant of support vector machines (SVMs) that use kernel functions to find the maximum-margin hyperplane in non-linear classification or regression problems. In simple terms, a kernel function transforms the original data into a higher-dimensional space, where it becomes linearly separable. The maximum-margin hyperplane is then found in this higher-dimensional space using an SVM.

Kernel SVMs have several advantages over regular SVMs. They can handle nonlinear classification or regression tasks without having to explicitly perform the data transformation, which can be computationally expensive. They also allow the use of different kernel functions, which can provide different types of non-linear transformations and can be chosen based on the specific characteristics of the data.

The most commonly used kernel functions in kernel SVMs are the linear, polynomial, and radial basis function (RBF) kernels. The linear kernel is used for linear classification or regression tasks, the polynomial kernel can handle nonlinear problems, and the RBF kernel is often used in classification tasks with a large number of features.

Kernel SVMs have been widely used in many applications, such as image and text classification, protein classification, and regression problems. However, they can be computationally expensive and may not be suitable for very large datasets. Additionally, choosing the right kernel function and the corresponding hyperparameters can be challenging and requires some domain knowledge and experimentation.



```
[ ] 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

Accuracy Score

```
[ ] # 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)
```

```
Accuracy score of the training data :  0.7866449511400652
```

```
[ ] # accuracy score on the test data  
X_test_prediction = classifier.predict(X_test)  
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
[ ] print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7727272727272727
```

## Accuracy\_score():

The `accuracy_score()` method of `sklearn.metrics`, accepts the true labels of the sample and the labels predicted by the model as its parameters and computes the accuracy score as a float value, which can likewise be used to obtain the accuracy score in Python. There are several helpful functions to compute typical evaluation metrics in the `sklearn.metrics` class. Let's use `sklearn's` `accuracy_score()` function to compute the Support Vector Classification model's accuracy score using the same sample dataset as earlier.

```
sklearn.metrics.accuracy_score(y_true, y_pred, *, normalize=True,  
sample_weight=None)
```

We use this for computing the accuracy score of classification. This method calculates subgroup accuracy in multi-label classification; a dataset's predicted subset of labels must precisely match the actual dataset of labels in `y_true`.

**Parameters:**

1. `y_true` (1d array-like, or array indicating label / sparse matrix): These are the true labels for a given sample.
2. `y_pred` (1d array-like, or array indicating label / sparse matrix): Predicted labels that a classification model has returned.
3. `normalize` (bool, default = True): It gives the number of successfully classified predicted samples if the answer is False. Returns the proportion of correctly classified predicted samples if True.
4. `sample_weight` (array-like of shape (n,), default = None): Sample weights.

**Returns**

1. `score` (float): It gives the ratio of successfully classified samples (float) if `normalize == True`; otherwise, it returns the count of successfully classified predicted samples (int). 1 is 100% accuracy for `normalize == True` and the count of samples provided with `normalize == False`.

From the above code, first store the svm classifier into a variable called `classifier` and then pass the `X_train` and `Y_train` variables into the `fit()` inbuilt function of `classifier`, then using the `predict()` function we pass the `X_train` data and train the model, for checking the accuracy score of the training data create a variable that accepts the accuracy score function pass the `X_train_prediction` variable that is already trained and check with the `Y_train` Outcome values. The accuracy score for training data is 0.78. That is a good score and the main part comes when training with the test data and for that the accuracy score is 0.77 which is also good because we didn't encounter any overfitting because the training and the testing accuracy score was nearly a match.

## Predictive Analytics:

What is Predictive Analytics?

Predictive Analytics is the use of mathematical and statistical methods, including artificial intelligence and machine learning, to predict the value or status of something of interest. Predictive analytics can be used to answer a vast array of questions such as:

- Which asset will fail within the next 75 days?
- How many units of product X will be sold next month?
- How much energy will a specific customer require in the next 15 minutes?
- Which customers will likely churn in the next 30 days?
- Is this transaction fraudulent?
- Is this image a car?
- Predicting a disease whether the person has it or not?

The benefits of predictive analytics are wide ranging and potentially game changing for a company. In particular, predictive analytics powered by AI/machine learning embedded in business processes can drive significant improvements such as reduced costs, increased margins and profitability, better safety and reliability, and lower environmental impact. A classic use of AI-powered predictive analytics enabling companies to reduce downtime, increase production, and reduce potential environmental impact (e.g., by avoiding leaks).

Why is Predictive Analytics Important?

Predictive analytics is a core function of AI. The foundational purpose of enterprise AI applications is to serve as “prediction engines,” providing insights to drive actions that improve business operations and performance. In the case of AI-powered predictive maintenance, for example, these applications can alert equipment and facilities operators in advance of critical failures, so that assets can be serviced or replaced before the failure – resulting in less unscheduled maintenance, fewer production and revenue interruptions, and increased asset life and utilisation.

There are limitless opportunities to apply predictive analytics in virtually every area of an organisation's operations. Machine learning models can be applied to customer interactions (for use cases such as product recommendations), financial transactions (for example, to predict instances of money laundering, predicting diseases), production (how much should be produced to meet predicted demand), and many other use cases.

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

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

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

# standardise 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')
```

Output:

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

From this above code, passing the numerical data of a single individual as input\_data that have all the values from columns in our dataset except Outcome value. Then converting into a numpy array and reshaping the array values for

standardisation then we feed the data into our classifier model that predicts the outcome if the person is diabetic or not.

[Link for the source code of the predictive model](#)

## **Conclusion:**

The changes offered by technological innovation have generated an extraordinary level of data collection and processing that is destined to undergo further expansion with the new applications of robotics and AI, crossing a new frontier and entering the era of big data and cognitive systems. A new category of technologies is born, which uses natural language processing and machine learning and is able to amplify and accelerate the digital transformation process to allow people and machines to interact in a more natural way, extending and enhancing cognitive skills and abilities. The possibility of extracting information that has a meaning and is functional, in fact, requires the development of sophisticated technologies and interdisciplinary skills to operate closely together. In particular, in medicine, health care systems require consistent, appropriate, and precise choices.

Today, the complexity of medicine certainly goes beyond the capacity of the human mind. The patients themselves are increasingly complex and we know that the long-term effectiveness of a treatment depends on variables that are no longer just “numerical,” but also on other information that is difficult to structure. In this framework, advances in computing power play a central role in the acquisition of knowledge. It is essential to collect and use the key information in a coherent way, out of its abundance by using effective and reliable analysis tools that are represented by the new techniques of AI already available today. These techniques recognize and use machine learning systems that are able to “extricate themselves” and learn from these immense amounts of data, even with intrinsic systems of recognition and error management. In essence, AI is a machine capable of solving problems and reproducing activities typical of human intelligence.

Our model’s accuracy of 77 out of 100 is like hitting the moon and promising, but it’s not too bad. Because of the limited data to train on the model leads to this accuracy that we find out when analysing our model and will be doing fine-tuning

and optimizations to improve the models performance in the future and we believe it will be a good model to predict if the person is diabetic or not.

## References:

- <https://www.geeksforgeeks.org/what-is-standardization-in-machine-learning/>
- <https://www.kaggle.com/code/mbalvi75/08-knn-diabetes-dataset/notebook>
- <https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing>
- Razavian N, Blecker S, Schmidt AM, Smith-McLallen A, Nigam S, Sontag D. Population-Level Prediction of Type 2 Diabetes From Claims Data and Analysis of Risk Factors. Big Data. 2015 Dec;3(4):277–87. doi: 10.1089/big.2015.0020.[[Google Scholar](#)]
- <https://www.who.int/news-room/fact-sheets/detail/diabetes>
- <https://scikit-learn.org/stable/modules/preprocessing.html#standardization-or-mean-removal-and-variance-scaling>