Department of Electronics and Communication

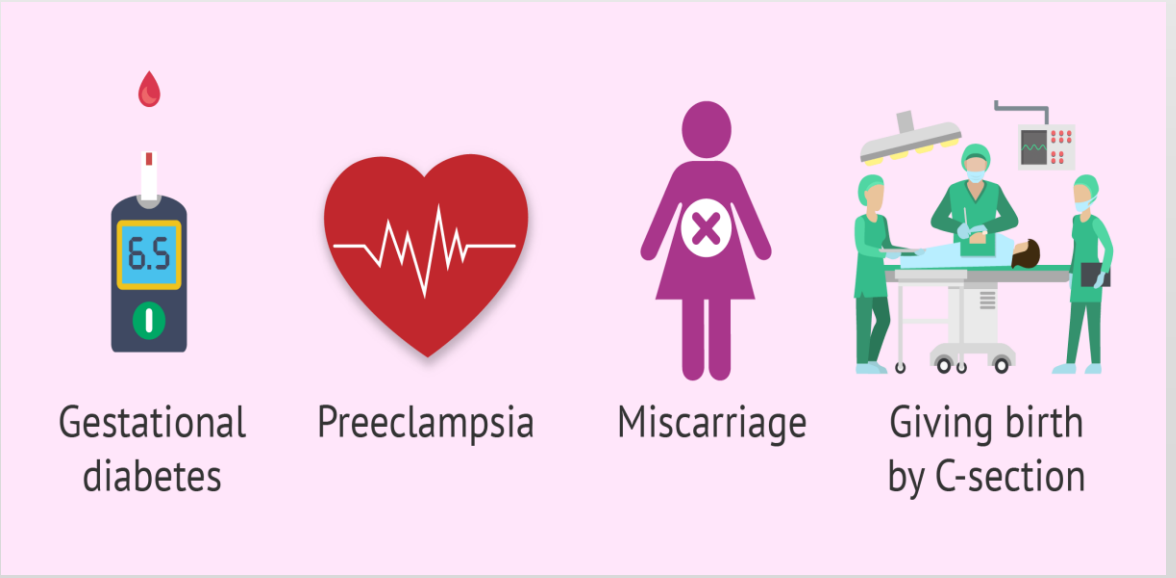
# PREDICT THE POSSIBLE HEALTH ISSUES IN PREGNANT WOMEN

Aaron lawrence lasrado - 4SO21EC002

# INTRODUCTION

According to WHO, 810 women die every day in this world due to childbirth and pregnancy-related complications, while the majority (94%) of all maternal deaths occur in low and lower-middle-income countries . Due to the recent advancement in technology, the rate of maternal deaths is reducing , yet is a challenging task to ensure the safety of both mother and child during pregnancy. In such a scenario, the pregnancy-related risks can be reduced by forecasting the complications and by taking preventive measures. Thus, the use of predictive modeling became emergent to save the lives of millions of mothers and infants.

The common maternal complications responsible for the majority of maternal deaths are gestational diabetes, severe bleeding, infection, preeclampsia, eclampsia, prolonged labor, preterm labor, and unsafe abortion .

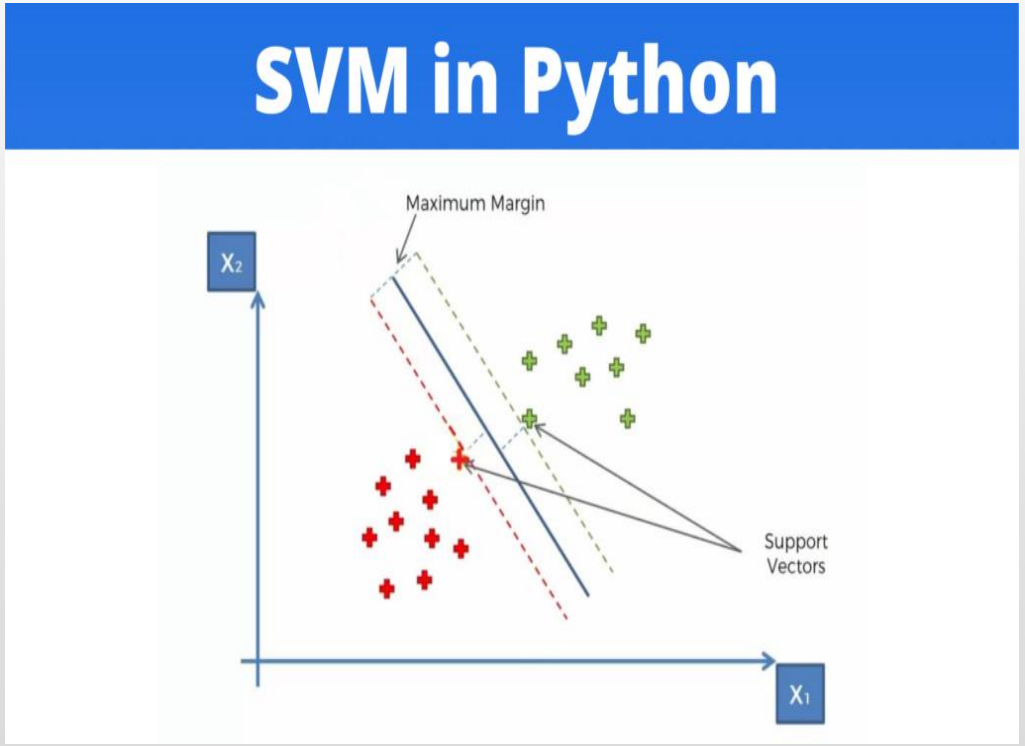


# Algorithm

## Support Vector Regression (SVR):

A Support Vector Regression (SVR) is a type of Support Vector Machine , and is a type of supervised learning algorithm that analyzes data for regression analysis.

In 1996, this version of SVM for regression was proposed by Christopher J. C. Burges, Vladimir N. Vapnik , Harris Drucker, Alexander J. Smola and Linda Kaufman. The model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.



# Implementation of program based on Algorithm

- Imported the packages /libraries to make it easier to write the program.

```
#Import the libraries
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
import matplotlib.pyplot as plt
import seaborn as sns
```

- Load the dataset.

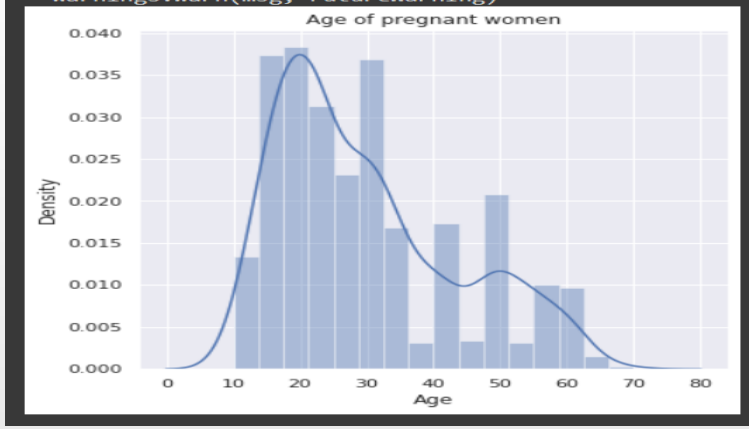
```
#Load the data
#from google.colab import files # Use to load data on Google Colab
#uploaded = files.upload() # Use to load data on Google Colab
pregnancy_dataset = pd.read_csv('/content/bootcamp.csv')
pregnancy_dataset.head()
```

| | Age | SystolicBP | DiastolicBP | BS | BodyTemp | HeartRate | RiskLevel |
|---|---|---|---|---|---|---|---|
| 0 | 25 | 130 | 80 | 15.0 | 98.0 | 86 | 1 |
| 1 | 35 | 140 | 90 | 13.0 | 98.0 | 70 | 1 |
| 2 | 29 | 90 | 70 | 8.0 | 100.0 | 80 | 1 |
| 3 | 30 | 140 | 85 | 7.0 | 98.0 | 70 | 1 |
| 4 | 35 | 120 | 60 | 6.1 | 98.0 | 76 | 0 |

- Plotted the models on a graph to see which has the best fit and returned the prediction

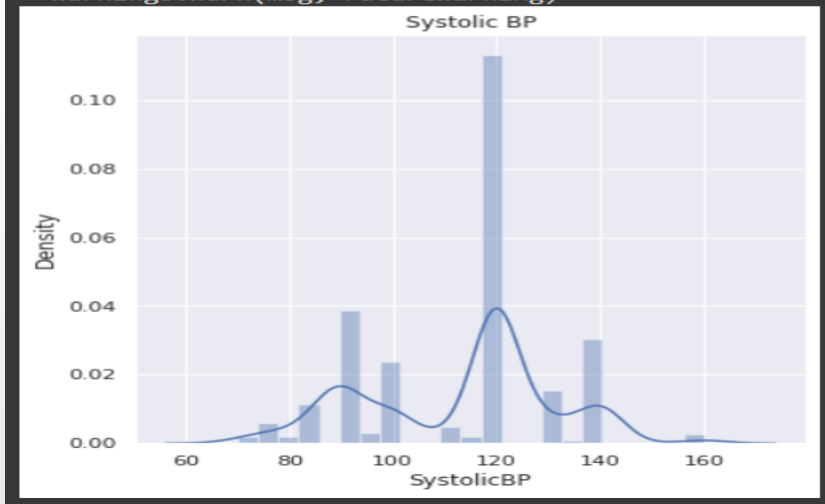
```
#Plot the models on a graph
sns.set()
plt.figure(figsize=(6,6))
sns.distplot(pregnancy_dataset['Age'])
plt.title('Age of pregnant women')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: Future
  warnings.warn(msg, FutureWarning)
```

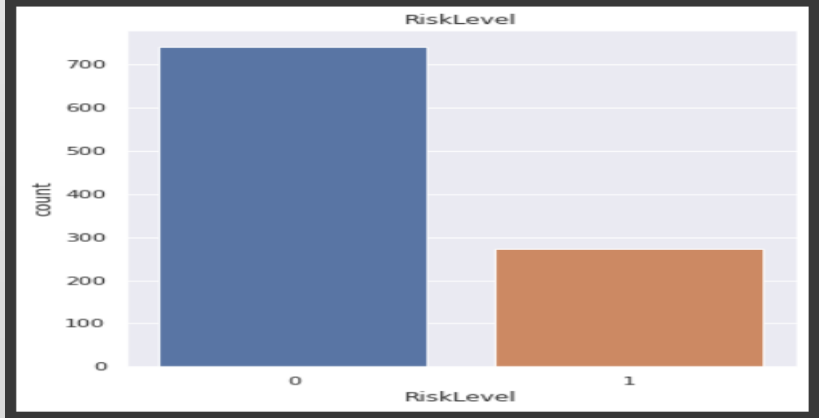



```
plt.figure(figsize=(6,6))
sns.distplot(pregnancy_dataset['SystolicBP'])
plt.title('Systolic BP')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: Futur
  warnings.warn(msg, FutureWarning)
```




```
plt.figure(figsize=(6,6))
sns.countplot(x='RiskLevel', data=pregnancy_dataset)
plt.title('RiskLevel')
plt.show()
```



# ➢ Splitting our Data into Testing and Training Data

```
#Splitting our Data into Testing and Training Data
X = pregnancy_dataset.drop(columns = 'RiskLevel', axis=1)
Y = pregnancy_dataset['RiskLevel']
print(X)
```

```
      Age  SystolicBP  DiastolicBP     BS  BodyTemp  HeartRate
0      25         130           80   15.0      98.0         86
1      35         140           90   13.0      98.0         70
2      29          90           70    8.0     100.0         80
3      30         140           85    7.0      98.0         70
4      35         120           60    6.1      98.0         76
...   ...         ...          ...    ...       ...        ...
1009   22         120           60   15.0      98.0         80
1010   55         120           90   18.0      98.0         60
1011   35          85           60   19.0      98.0         86
1012   43         120           90   18.0      98.0         70
1013   32         120           65    6.0     101.0         76

[1014 rows x 6 columns]
```

```
print(Y)
```

```
0       1
1       1
2       1
3       1
4       0
       ..
1009    1
1010    1
1011    1
1012    1
1013    0
Name: RiskLevel, Length: 1014, dtype: int64
```

+ Code + Text

```
scaler = StandardScaler()
scaler.fit(X)
```

```
StandardScaler()
```

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

```
[[-0.36173812  0.91339632  0.25502279  1.90589019 -0.4852155   1.44695615]
 [ 0.38077697  1.45702716  0.97553854  1.29833966 -0.4852155  -0.53208757]
 [-0.06473208 -1.26112705 -0.46549297 -0.22053665  0.97388449  0.70481475]
 ...
 [ 0.38077697 -1.53294248 -1.18600873  3.12099124 -0.4852155   1.44695615]
 [ 0.97478904  0.36976548  0.97553854  2.81721597 -0.4852155  -0.53208757]
 [ 0.15802244  0.36976548 -0.82575085 -0.82808717  1.70343448  0.21005383]]
```

```
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)
```

```
(1014, 6) (811, 6) (203, 6)
```

```
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train, Y_train)
```

```
SVC(kernel='linear')
```

- Building and training our model

```
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 :  0.8606658446362515
```

- Building and testing our model

```
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 :  0.8374384236453202
```

We can now use the model to make predictions of the data. We can do this by using the .predict() method and passing in our testing features.

## ➢ Input module for the prediction model

```
# Input module
input_data = (55,110,85,6.9,98,88)

# 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 pregnant patient is at low risk')
else:
  print('The pregnant patient is at high risk')
```

```
# Input module
input_data = (35,140,90,13,98,70)

# 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 pregnant patient is at low risk')
else:
  print('The pregnant patient is at high risk')
```

## ➢ Output

```
[[ 1.86580715 -0.17386537  0.61528066 -0.55468943 -0.4852155   1.69433661]]
[0]
The pregnant patient is at low risk
```

```
[[ 0.38077697  1.45702716  0.97553854  1.29833966 -0.4852155  -0.53208757]]
[1]
The pregnant patient is at high risk
```

# THANK YOU

Reference:

https://www.youtube.com/watch?v=xUE7SjVx9bQ&t=722s
https://colab.research.google.com/drive/1oxnhMTlomJ4HVhPuowpPFyMt1mwuOuQo?usp=sharing
https://datagy.io/python-support-vector-machines/
https://archive.ics.uci.edu/ml/machine-learning-databases/00639/Maternal%20Health%20Risk%20Data%20Set.csv