Objective:

This Project involves applying supervised learning techniques, specifically **Logistic Regression** and **Support Vector Machines** (SVM), to the C-stick Fall Prediction Dataset. You can read about the dataset here. You will then perform predictions On the data using each model, then finally you will compare the performance of these two algorithms and interpret their efficacy in making predictions based on the data.

Steps

This Project is divided into 2 parts: **Section 1** and **Section 2**. Section 1 is implemented for you, and it involves loading the data, loading the necessary libraries and the separation of the data in features (inputs/attributes/Predictor variables) and labels (output/target/Predicted variables). You will implement Section 2, and it involves the following tasks:

1. Split the data into training sets and test sets
2. Implement the SVM Model
3. Implement predictions on the SVM model using an accuracy metric
4. Implement the Log Reg Model
5. Implement predictions on the Log Reg Model an accuracy metric

```
     # -*- coding: utf-8 -*-
"""fall-prediction-using-ml.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1T9CbfJCYEVTJ_E26hO7H9SQtujtMeh4V
"""
#Training the model:
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

df_Cstick = pd.read_csv("/content/cStick.csv")
print(df_Cstick)
'''Output:

      Distance  Pressure      HRV  Sugar level    SpO2  Accelerometer   Decision
0       25.540       1.0  101.396       61.080  87.770            1.0       1
1        2.595       2.0  110.190       20.207  65.190            1.0       2
2       68.067       0.0   87.412       79.345  99.345            0.0       0
3       13.090       1.0   92.266       36.180  81.545            1.0       1
4       69.430       0.0   89.480       80.000  99.990            0.0       0
...        ...       ...      ...          ...     ...            ...       ...
```

```
2034     5.655       2.0  116.310      162.242  71.310           1.0       2
2035     9.660       2.0  124.320      177.995  79.320           1.0       2
2036    15.220       1.0   93.828       40.440  82.610           1.0       1
2037     9.120       2.0  123.240      175.871  78.240           1.0       2
2038    62.441       0.0   78.876       76.435  96.435           0.0       0
[2039 rows x 7 columns]

'''
#############################################################################################

# Separate features and labels
features = df_Cstick[['Distance', 'Pressure', 'HRV', 'Sugar level', 'SpO2',
'Accelerometer']]
print(features)




'''Output:

      Distance  Pressure      HRV  Sugar level    SpO2  Accelerometer
0       25.540       1.0  101.396       61.080  87.770           1.0
1        2.595       2.0  110.190       20.207  65.190           1.0
2       68.067       0.0   87.412       79.345  99.345           0.0
3       13.090       1.0   92.266       36.180  81.545           1.0
4       69.430       0.0   89.480       80.000  99.990           0.0
...        ...       ...      ...          ...     ...           ...
2034     5.655       2.0  116.310      162.242  71.310           1.0
2035     9.660       2.0  124.320      177.995  79.320           1.0
2036    15.220       1.0   93.828       40.440  82.610           1.0
2037     9.120       2.0  123.240      175.871  78.240           1.0
2038    62.441       0.0   78.876       76.435  96.435           0.0

[2039 rows x 6 columns]
'''
#############################################################################################

df_Cstick = df_Cstick.rename(columns={"Decision ": "Decision"})
labels = df_Cstick['Decision']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2,
random_state=42)

# Create and fit SVM model
svm_model = SVC()
svm_model.fit(X_train, y_train)

# Predict on the test set
```

```
y_pred = svm_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print('The accuracy of the SVM Model is:', accuracy)

'''
Output:
The accuracy of the SVM Model is: 1.0

'''
##########################################################################################
```

```
#Testing using the svm model

import joblib

# Save the trained SVM model to a .pkl file
joblib.dump(svm_model, 'svm_model.pkl')

from google.colab import files

files.download('svm_model.pkl')

svm_model = joblib.load('/content/svm_model.pkl')

# Define a function to predict fall based on sensor data
def predict_fall(sensor_data):
    # Make prediction using the trained SVM model
    prediction = svm_model.predict([sensor_data])[0]
    if prediction == 2:
        return 'Take care of your child/patient.'
    else:
        return 'No significant abnormal variations detected.'

# Generate some random sensor data for testing
random_sensor_data = [2.6, 3, 111, 20.207, 66, 1.0]  # Replace with your random sensor
data

# Test the model with the random sensor data
prediction = predict_fall(random_sensor_data)
print(prediction)

'''
```

```
Output:
Take care of your child/patient.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have
valid feature names, but SVC was fitted with feature names
  warnings.warn(

'''
```