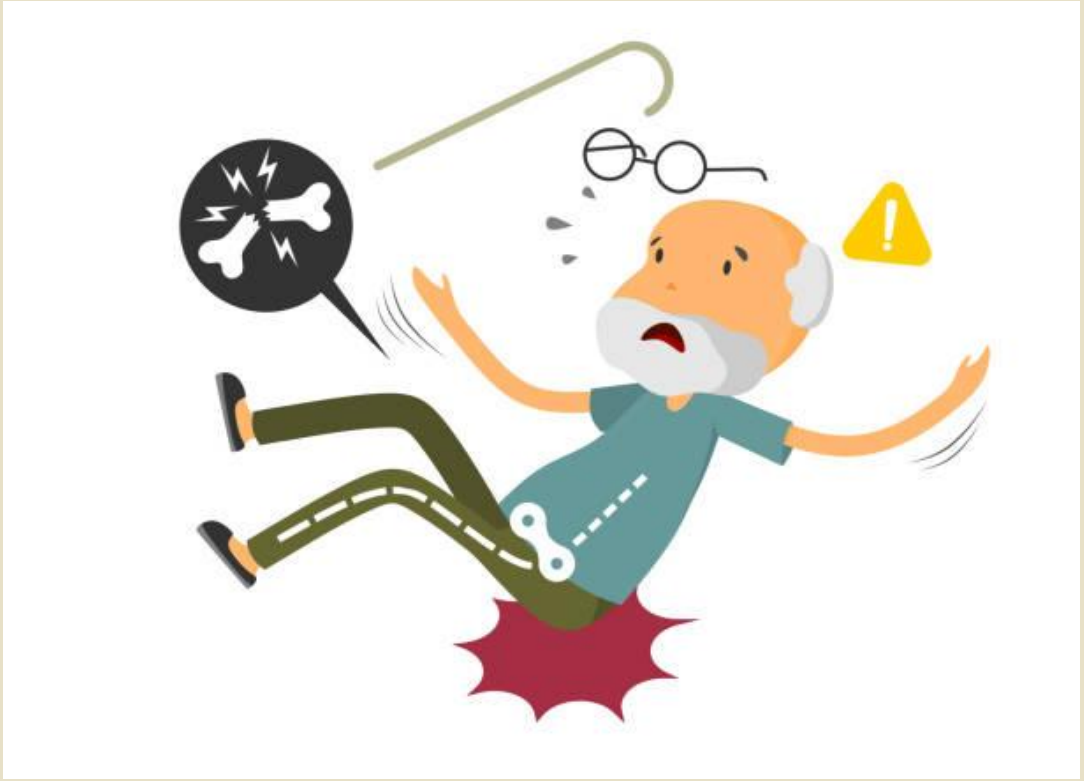
FALL PREDICTION AND DETECTION USING RANDOM FOREST ALGORITHM

GROUP ID - P5005

ABSTRACT:

- Falls are constant threats to older adults and can minimize their ability to live independently.
- To make life easier for elder adults who may have visual or hearing impairments our model is proposed.



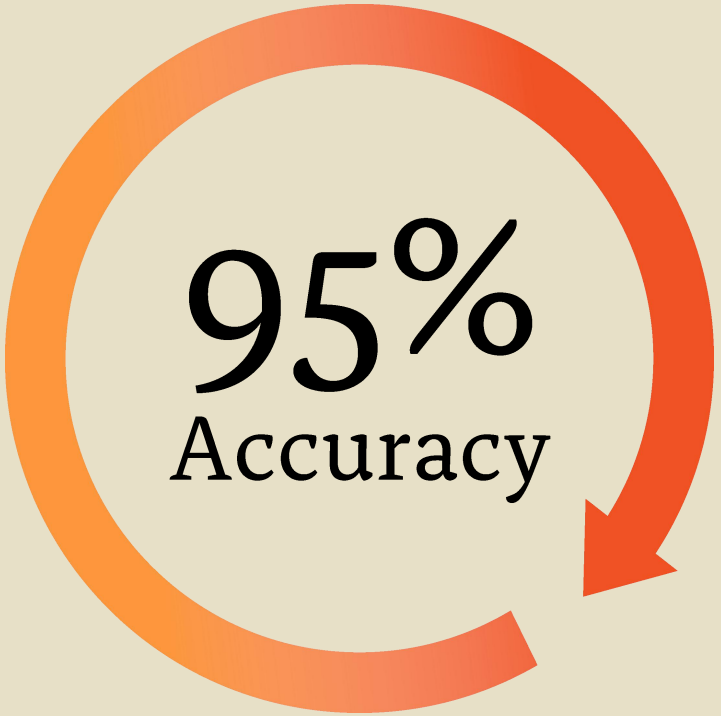
ABSTRACT :

- Our model capability to predict falls before their occurrence, to warn the user that there may be an incident of fall, to detect falls to reduce their impact.

- Our model monitors , the physiological features that occur when a person is about to fall and also monitors the surroundings the user is in when having an incident of fall.

ABSTRACT:

- Based on the changes in the monitored parameters, the decision of fall i.e., a prediction, warning or detection of fall is made with an accuracy of approximately 95%.



INTRODUCTION:

- The impact of falls on the elderly human body is very high. Falls can cause broken bones, wrists, arms, ankles and hips.

- In order to reduce such accidents, our model is to monitor falls in older adults is proposed. Our model has an ability to not only detect falls but to also predict the incident of fall so as to reduce their occurrences.

INTRODUCTION:

- We use random forest algorithm to create this model which gives the best of best prediction and detection of falls. And use cross validation techniques for optimising the accuracy .

- Our aim is to create a model which can not only detect falls which can accurately predict falls and reduces accurance of falls.

RELATED PRIOR RESEARCH:

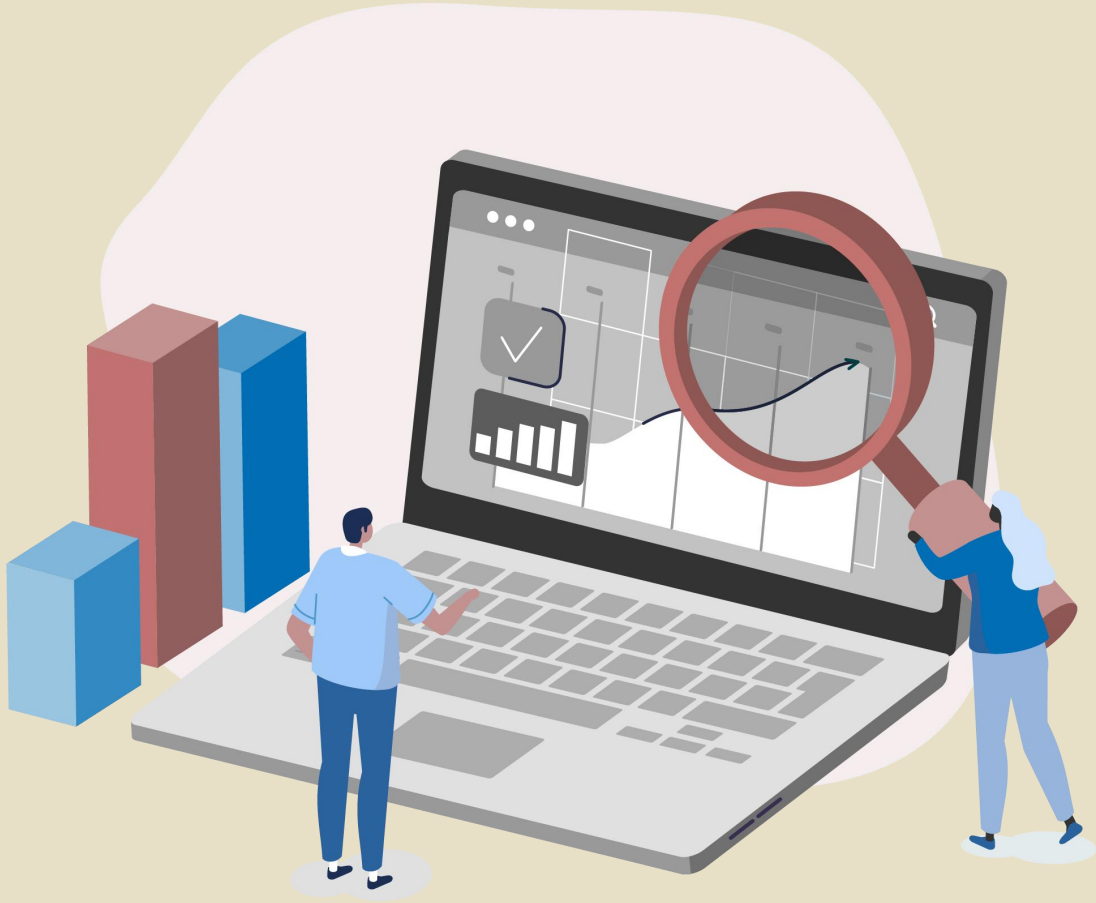
- Though there is a good number of models in the market that provide care for older adults, most of them only focus on fall detection and not prediction.

- Some of the major disadvantages of these models are the generation of false positives and false negatives which can cause unnecessary alerts, monthly maintenance charges and privacy concerns.

RELATED PRIOR RESEARCH:

Some of the major issues with the existing solutions are:

– There are no fall prediction mechanisms or strategies included.



IMPLEMENTATION

Importing the requires librariers:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/content/Project (2).csv'):
    for filename in filenames:
      print(os.path.join(dirname, filename))
```

Features List:
Distance , Pressure ,HRV , Blood Sugar, SPo2
and accelerometer readings.

DATA ACQUISTION PARAMETER RANGE FOR FALL DETECTION AND PREDICTION IN OUR MODEL:

- For implementation and validation in our model a dataset of 2039 sample based on table given was used.

Distance	Pressure	HRV	Sugar Levels	SpO2 Levels	Accelerometer	Decisions
> 50cm	Small	60- 90bpm	70-80 mg/dL	> 90	< Threshold	No fall. Happy walking!
< 30cm	Medium	90- 105bpm	30-70 mg/dL	80-90	> Threshold	Take a break, you tripped!
< 10cm	Large	>105 bpm	< 30 mg/dL or > 160 mg/dL	<80	> Threshold	Definite fall. Help is on the way!

DATA SPLITTING:

```
[27] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

RANDOM FOREST CLASSIFER:

The logic behind the Random Forest model is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone. When using Random Forest for classification, each tree gives a classification or a "vote." The forest chooses the classification with the majority of the "votes." When using Random Forest for regression, the forest picks the average of the outputs of all trees.

- When using a regular decision tree, you would input a training dataset with features and labels and it will formulate some set of rules which it will use to make predictions. If you entered that same information into a Random Forest algorithm, it will randomly select observations and features to build several decision trees and then average the results.
- Implementation of random forest algorithm.

```
from sklearn import tree
from sklearn import ensemble

# chose random values for depth and number of trees
depth =1
num_trees = 4

#Build model
# Use max_samples=0.5. This is the fraction of rows to use for each DT
# For all of our forests, we will let max_samples be 0.5   We'll explore best_d and best_num_trees..
clf = ensemble.RandomForestClassifier(max_depth = depth, n_estimators = num_trees, max_samples = 0.5)
#Train Model
clf.fit(X_train,y_train)
print(f"Built an RF with depth={depth} and number of trees={num_trees}")
```

```
Built an RF with depth=1 and number of trees=4
```

We use accuracy score to measure the accuracy of the model before cross validation :

```
[ ]  from sklearn.metrics import accuracy_score

     predicted_labels = clf.predict(X_test)
     actual_labels = y_test

     accuracy = accuracy_score(actual_labels,predicted_labels)
     print(f'This model is {accuracy} or {accuracy*100:7.4f}% accurate')
```

```
This model is 0.6200980392156863 or 62.0098% accurate
```

- Cross Validation

```
from sklearn.model_selection import cross_val_score
best_d = 1         # range(1,6)
best_ntrees = 1   # [50,150,250]
best_accuracy = 0

for d in range(1,6):
    for n_trees in [2,3,4]:
        clf = ensemble.RandomForestClassifier(max_depth=d,n_estimators=n_trees,max_samples=0.5)
        cv_scores = cross_val_score(clf,X_train,y_train,cv=5) # 5 means 80/20 split
        average_cv_accuracy =cv_scores.mean()
        print(f"depth: {d:2d} ntress: {n_trees: 3d} cv accuracy: {average_cv_accuracy:7.4f}")
        if average_cv_accuracy>best_accuracy:
            best_accuracy = average_cv_accuracy
            best_d = d
            best_ntrees = n_trees

best_depth = best_d
best_num_trees = best_ntrees

print(f"best_depth: {best_depth} and best_num_trees: {best_num_trees} are our choices. Acc: {best_accuracy}")
```

After Cross validation the accuracy score will be

```
[31] #Build Model
clf_validated = ensemble.RandomForestClassifier(max_depth = best_depth, n_estimators = best_num_trees, max_samples = 0.5)

#Train model
clf_validated.fit(X_train,y_train)

#Test model
predicted_labels = clf_validated.predict(X_test)
actual_labels = y_test

accuracy = accuracy_score(actual_labels,predicted_labels)
print(f'This model is {accuracy} or {accuracy*100:7.4f}% accurate')
```

```
This model is 1.0 or 100.0000% accurate
```

- Now its time for prediction:

```
[33] def predictive_model(Features,CLF):
        """ input 1: a list of 4 project features:
            [ 'Distance', 'Pressure', 'HRV', 'Sugar level', 'SpO2', 'Accelerometer']

            input 2 : Classifier/model CLF

            output: outcome/Decision from:
            0:'No Fall detected'
            1:'Slip detected'
            2:'Definite fall'
        """
        our_features = np.asarray([Features])                # extra brackets needed
        predicted_status = CLF.predict(our_features)
        # force to int from float so we can use for indexing purposes (among other things)
        predicted_status = int(predicted_status)
        outcome = outcome_list[predicted_status]

        return outcome
```

Predicted on new and unseen data:

```
# List of features, Lof  (recall  features [ 'Distance', 'Pressure', 'HRV', 'Sugar level', 'SpO2', 'Accelerometer'])


LoF =  [ [28,1,123.24,45.89,96.456,1.0],
         [45,0,82.61,0.456,98.05,0.0],
         [28,1,123.24,200,4.6,1.0],
         [2,9,78.876,23.44,53.2,0.0],
         [80,1,23.24,175.8,71,1.0],
         [63,0,13.24,6.46,9.3,0.0]]


for feature in LoF:
    decision = predictive_model(feature,clf_RF)
    print(f'The MODEL gave the decision of a "{decision}" for the features: {feature}')
```

```
The MODEL gave the decision of a "Slip detected" for the features: [28, 1, 123.24, 45.89, 96.456, 1.0]
The MODEL gave the decision of a "No Fall detected" for the features: [45, 0, 82.61, 0.456, 98.05, 0.0]
The MODEL gave the decision of a "Slip detected" for the features: [28, 1, 123.24, 200, 4.6, 1.0]
The MODEL gave the decision of a "No Fall detected" for the features: [2, 9, 78.876, 23.44, 53.2, 0.0]
The MODEL gave the decision of a "Slip detected" for the features: [80, 1, 23.24, 175.8, 71, 1.0]
The MODEL gave the decision of a "No Fall detected" for the features: [63, 0, 13.24, 6.46, 9.3, 0.0]
```

CONCLUSION:

- Elderly care requires utmost attention with the technological developments happening day by day. Keeping that in mind, our model is proposed with which the users are not only provided with a fall detection monitoring device but also a fall prediction device with an accuracy of approximately 96.67%. The model also monitors the vital signals of the older adults and this allows in the early detection of accidents including falls.

CONCLUSION:

- Elderly care requires utmost attention with the technological developments happening day by day. Keeping that in mind, our model is proposed with which the users are not only provided with a fall detection monitoring device but also a fall prediction device with an accuracy of approximately 96.67%. The model also monitors the vital signals of the older adults and this allows in the early detection of accidents including falls.