PROJECT REPORT

TITLE: ACCURATE BODY FAT PREDICTION

1. Introduction

**1.1 Overview**

Body fat prediction plays a crucial role in various fields, including health and fitness, sports performance, and medical research. Accurate estimation of body fat percentage is important for assessing overall health and monitoring progress in weight management programs.

We are using Random Forest Regressor, which belongs to the family of ensemble learning methods. Random Forest is a powerful technique that combines multiple decision trees to create a robust and accurate prediction model.

**1.2 Purpose**

Body fat percentage is closely linked to various health conditions such as obesity, cardiovascular disease, and diabetes. Predicting body fat percentage helps in assessing an individual's risk for these health issues. Body fat prediction is valuable in weight management programs. By accurately estimating body fat percentage, individuals can set realistic goals and track progress effectively. Body fat prediction contributes to scientific research in fields such as nutrition, epidemiology, and public health. It allows researchers to analyze large datasets, investigate the relationship between body fat and other health parameters.

1. Literature survey

**2.1 Existing problem**

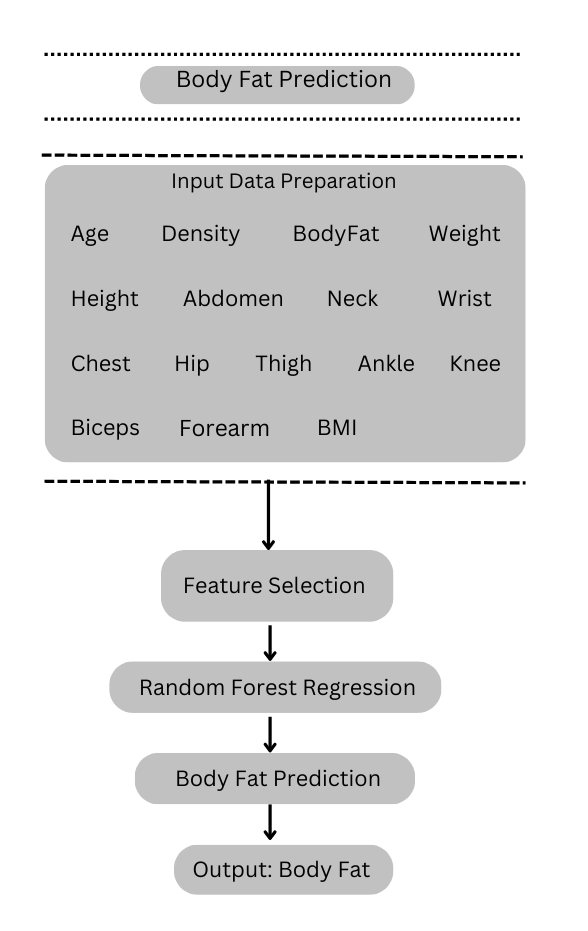
One of the existing problems in body fat prediction is the accuracy and reliability of the prediction models. Although various methods and algorithms have been developed to estimate body fat percentage, there can still be significant variations and discrepancies between the predicted values and actual measurements.

**2.2 Proposed solution**

We are proposing a solution by creating a model using Random Forest regressor. Random Forest has ability to handle high-dimensional datasets and automatically select the most relevant features. This eliminates the need for manual feature selection and reduces the risk of overfitting. We are removing the outliers using IQR then we are scaling the data using standard scaler. We were able to achieve an accuracy of 99.8

1. Theoritical analysis

**3.1 Block diagram**



**3.2 Hardware / Software designing**

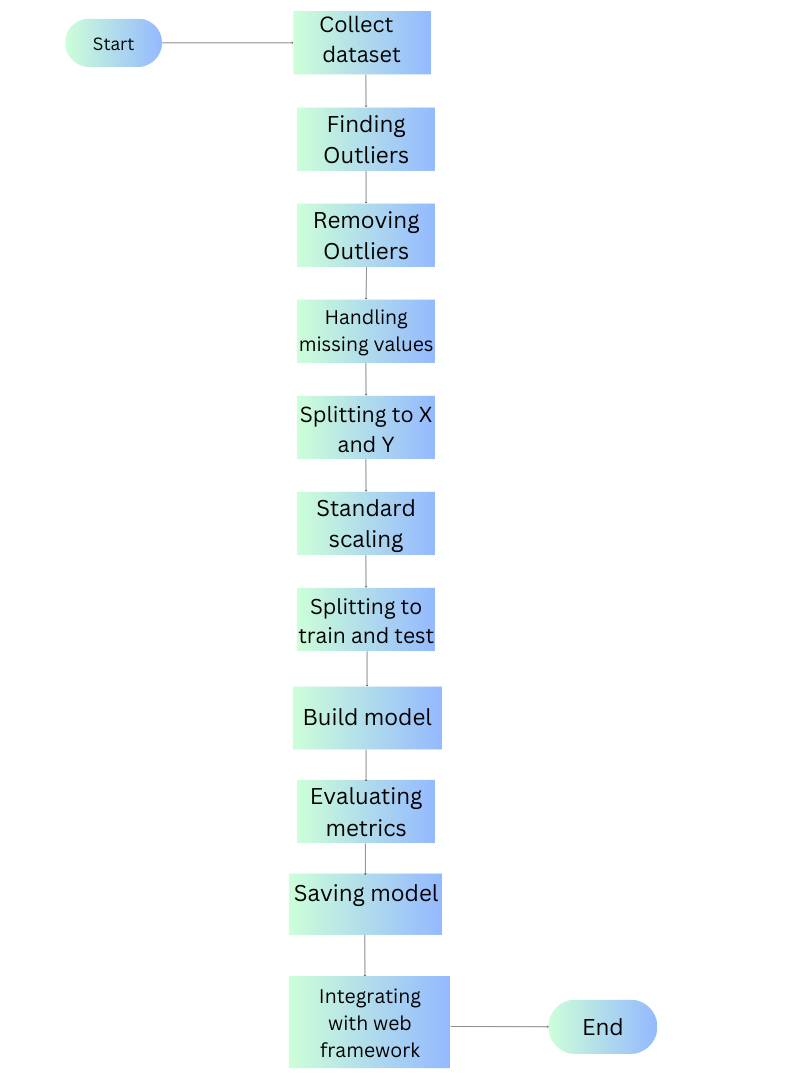
Hardware requirements: Computer or Server**,** Storage, Graphics Processing Unit, Memory (RAM)

Software requirements: Machine Learning Libraries, Data Preprocessing Tools, Development Environment like Jupyter notebook, Operating System

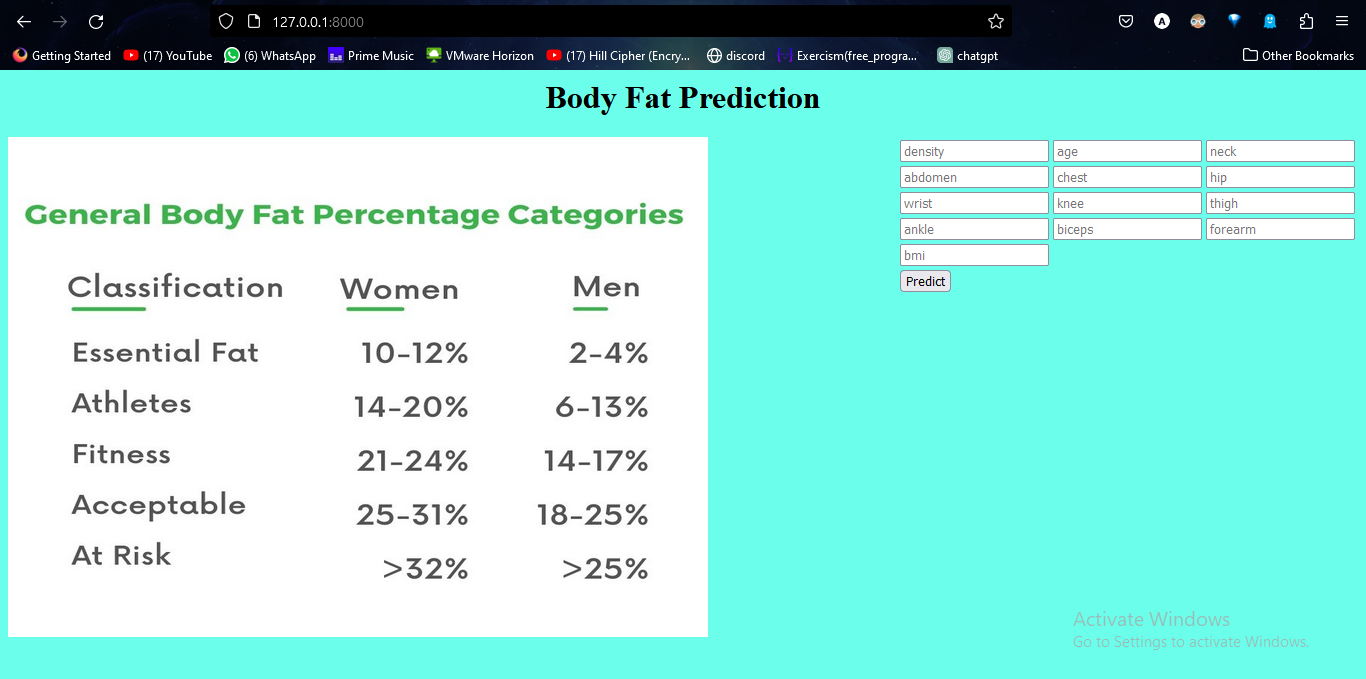
1. Experimental investigations

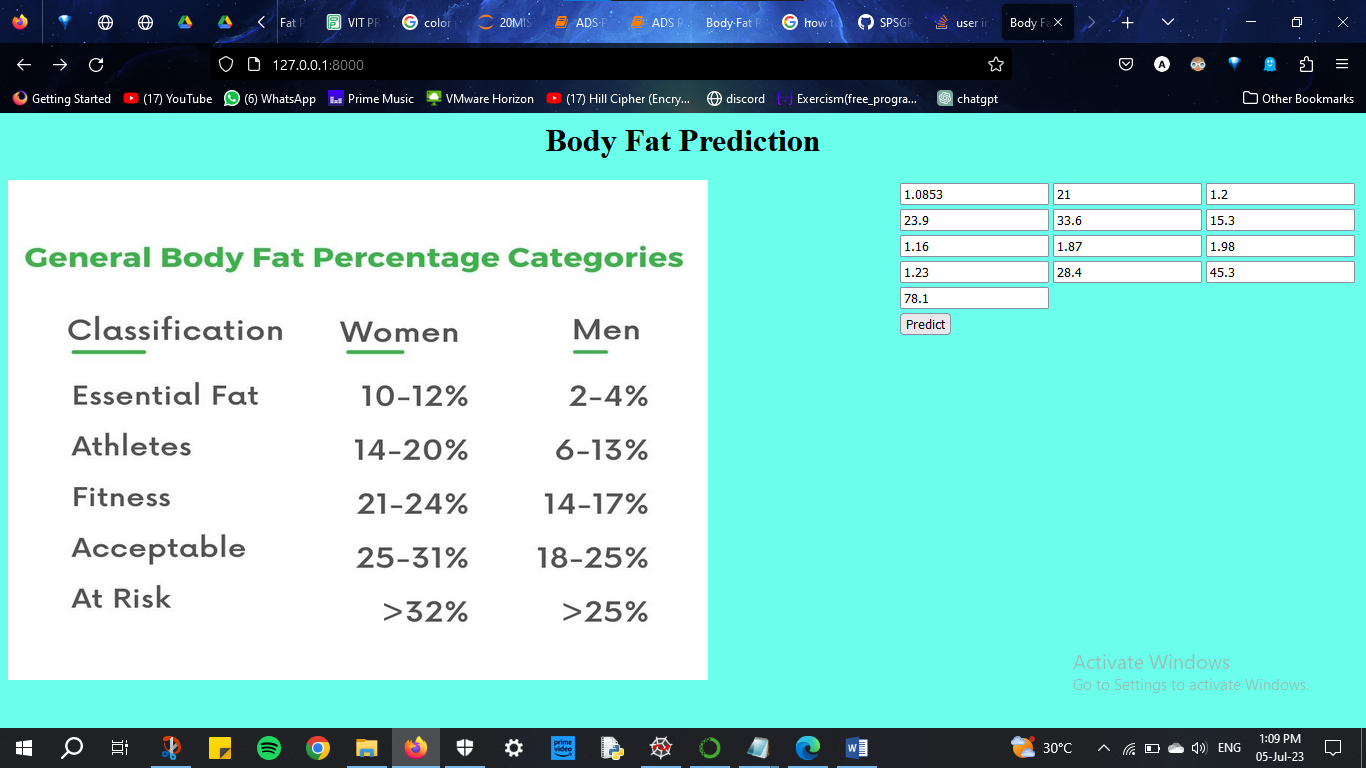
We have tried building the model using Linear Regression, Random Forest, Ridge and Lasso, Decision Tree Regressor and Knn. Out of all these algorithms we found that Random Forest gives the highest accuracy of 99.8%

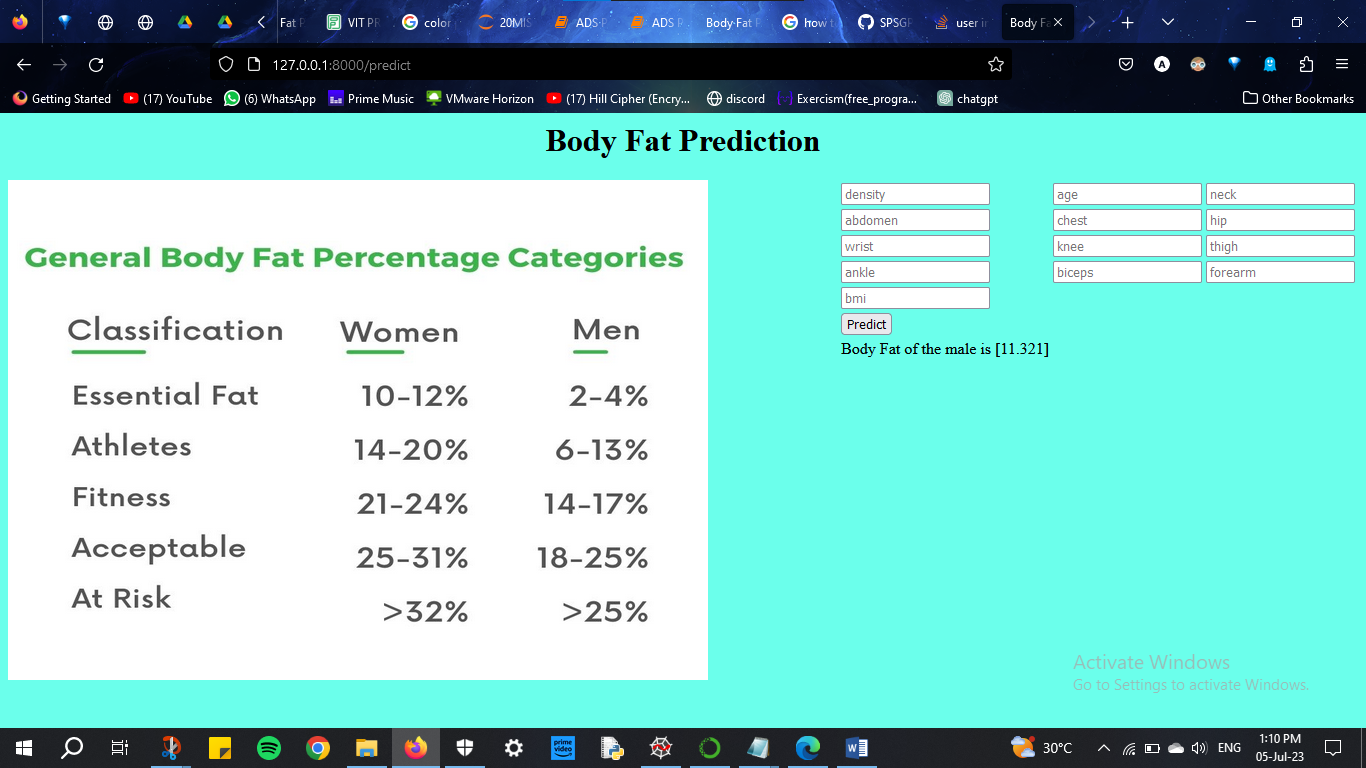
1. Flowchart



1. Result







1. Advantages and disadvantages

Advantages of using Random Forest Regression for body fat prediction:

* Non-linear Relationships: It can handle complex patterns and interactions in the data, making it suitable for capturing the intricacies of body fat prediction.
* Robustness: Random Forest models are robust to outliers and noise in the data.
* Feature Importance: Random Forest models provide a measure of feature importance, which can help identify the most influential factors in predicting body fat percentage.
* Scalability: Random Forest can handle large datasets efficiently.

Disadvantages of using Random Forest Regression for body fat prediction:

* Model Interpretability: It may be difficult to explain the relationships between input features and body fat percentage.
* Overfitting: Although Random Forest is less prone to overfitting compared to individual decision trees, it can still overfit if the number of trees in the forest is too large or the model is not properly tuned.
* Limited Extrapolation Capability: Random Forest models are primarily designed for interpolation rather than extrapolation. Therefore, predictions for values outside the range of the training data may not be as reliable.

1. Applications

Key Areas where Body Fat Prediction is applied:

* Health and Fitness: Body fat prediction is widely used in the health and fitness industry for assessing an individual's overall health and fitness levels.
* Sports and Athletics: Body fat percentage to monitor and optimize body composition for improved performance.
* Nutritional Assessment: Body fat prediction is used to assess nutritional status and guide dietary interventions.
* Clinical Settings: Body fat prediction is valuable in clinical settings for assessing patients' health conditions and guiding treatment plans.
* Research and Epidemiology: Body fat prediction is used in research studies and epidemiological surveys to analyze population health data.
* Body Image and Cosmetics: Body fat prediction is also utilized in the cosmetics and body image industries.

1. Conclusion

In our project we have predicted body fat using random forest regressor. We have deployed the model using Flask application.

1. Future scope

Further research can focus on identifying and incorporating additional relevant features that may contribute to more accurate body fat prediction. This can include exploring new types of data, such as genetic or biomarker data. The development of personalized body fat prediction models can be explored. This involves considering individual characteristics, such as age, gender, activity level, and lifestyle factors, to tailor the prediction specifically to an individual's profile.

1. Bibliography

<https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/>

<https://stackabuse.com/decision-trees-in-python-with-scikit-learn/>

<https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/>

<https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/>

APPENDIX

Source code:

#importing libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#dataset

data=pd.read\_csv('bodyfat.csv')

data.head()

data.describe()

data.info()

#Finding Outliers

plt.figure(figsize=(20,14))

sns.boxplot(data=data)

plt.show()

data.isnull().sum()

#adding BMI to data

data['Bmi']=703\*data['Weight']/(data['Height']\*data['Height'])

data=data.drop(columns=['Weight','Height'],axis=1)

data.head()

#Removing Outliers

q1=data.quantile(0.25)

q3=data.quantile(0.75)

IQR=q3-q1

data=data[~((data<(q1-1.5\*IQR)) | (data>(q3+1.5\*IQR)))]

data.describe()

data.isnull().sum()

sns.pairplot(data)

#handling missing values

data['Density'].fillna(data['Density'].mean(),inplace=True)

data['BodyFat'].fillna(data['BodyFat'].mean(),inplace=True)

data['Neck'].fillna(data['Neck'].mean(),inplace=True)

data['Chest'].fillna(data['Chest'].mean(),inplace=True)

data['Abdomen'].fillna(data['Abdomen'].mean(),inplace=True)

data['Hip'].fillna(data['Hip'].mean(),inplace=True)

data['Thigh'].fillna(data['Thigh'].mean(),inplace=True)

data['Knee'].fillna(data['Knee'].mean(),inplace=True)

data['Ankle'].fillna(data['Ankle'].mean(),inplace=True)

data['Biceps'].fillna(data['Biceps'].mean(),inplace=True)

data['Forearm'].fillna(data['Forearm'].mean(),inplace=True)

data['Wrist'].fillna(data['Wrist'].mean(),inplace=True)

data['Bmi'].fillna(data['Bmi'].mean(),inplace=True)

plt.figure(figsize=(20,14))

sns.boxplot(data=data)

plt.show()

#splitting to independent and dependent variables

Y=data['BodyFat']

X=data.drop(columns=['BodyFat'],axis=1)

#StandardScaler

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X)

inputs\_scaled = scaler.transform(X)

name=X.columns

X=pd.DataFrame(inputs\_scaled,columns=name)

X

#splitting to test and train data

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\_state=0)

X\_train.head()

X\_test.head()

print(X\_train.shape)

print(X\_test.shape)

print(Y\_train.shape)

print(Y\_test.shape)

#Building model using Linear Regression

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

from sklearn.linear\_model import LinearRegression

lr=LinearRegression()

lr.fit(X\_train,Y\_train)

Y\_pred=lr.predict(X\_test)

Y\_pred

E=Y\_test-Y\_pred

#evaluation metrics

from sklearn.metrics import r2\_score

acc=r2\_score(Y\_pred,Y\_test)

acc

#building model using Random Forest

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y, test\_size=0.2, random\_state=42)

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n\_estimators = 100, random\_state = 0)

rf.fit(X\_train, Y\_train)

pred=rf.predict(X\_test)

#evaluation metrics

from sklearn.metrics import r2\_score,mean\_squared\_error,mean\_absolute\_error

accuracy=r2\_score(Y\_test,pred)

accuracy

print('MAE: ', mean\_absolute\_error(Y\_test,pred))

print('MSE: ', mean\_squared\_error(Y\_test,pred))

#build model using Ridge and Lasso

from sklearn.linear\_model import Ridge

from sklearn.linear\_model import Lasso

r=Ridge()

l=Lasso()

r.fit(X\_train,Y\_train)

l.fit(X\_train,Y\_train)

pred1=r.predict(X\_test)

pred2=l.predict(X\_test)

#evaluation metrics

from sklearn import metrics

print(metrics.r2\_score(Y\_test,pred1))

print(metrics.r2\_score(Y\_test,pred2))

#MSE (Mean Square Error)

print(metrics.mean\_squared\_error(Y\_test,pred1))

print(metrics.mean\_squared\_error(Y\_test,pred2))

#RMSE(Root Mean Square Error)

print(np.sqrt(metrics.mean\_squared\_error(Y\_test,pred1)))

print(np.sqrt(metrics.mean\_squared\_error(Y\_test,pred2)))

#build model using decision tree

from sklearn.tree import DecisionTreeRegressor

df=DecisionTreeRegressor(random\_state=0)

df=df.fit(X\_train,Y\_train)

pred=df.predict(X\_test)

pred

#evaluation metrics

from sklearn.metrics import mean\_squared\_error, r2\_score

print('Mean Absolute Error:', metrics.mean\_absolute\_error(Y\_test,pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(Y\_test,pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(Y\_test,pred)))

r2 = r2\_score(Y\_test,pred)

r2

#building model using Knn

from sklearn.neighbors import KNeighborsRegressor

knn=KNeighborsRegressor()

knn.fit(X\_train,Y\_train)

pred=knn.predict(X\_test)

#evaluating metrics

r2\_score(Y\_test,pred)

from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error

from math import sqrt

mae = mean\_absolute\_error(Y\_test,pred)

mse = mean\_squared\_error(Y\_test,pred)

rmse = sqrt(mse)

mse

rmse

mae

from sklearn.ensemble import RandomForestRegressor

from sklearn.datasets import make\_regression

import joblib

# Generate a random regression dataset for demonstration

X,Y = make\_regression(n\_samples=100, n\_features=10, random\_state=42)

# Create and fit the Random Forest model

rf\_model = RandomForestRegressor()

rf\_model.fit(X,Y)

# Save the trained model to a file

joblib.dump(rf\_model, 'random\_forest\_model.pkl')

rf\_model.predict(X)

loaded\_model = joblib.load('random\_forest\_model.pkl')

TEAM NO: 521

TITLE: ACCURATE BODY FAT PREDICTION USING MACHINE LEARNING

NAME:

REG NO:

CAMPUS: