Height and Weight Predication using ML

Prepared For:
Abdullah Fiaz
Lab Instructor, Artificial Intelligence Lab – v1,
University of Management and Technology

By:

Minaam Ahmad	**F2021266555**
Azka Khalid	**F2021266571**

23 January, 2025

Abstract:
This project explores the use of facial landmark-based features for predicting height and weight of a human body. A dataset of facial images was annotated using the shape_predictor_68_face_landmarks model to extract key facial dimensions such as eye distance, nose-to-chin distance, and jaw width. Height and weight were manually added to the dataset as target variables. Machine learning models, including Linear Regression, Support Vector Machines, and Random Forest were trained on the extracted features to predict the target variables. The models were evaluated using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2). Results highlight the potential of facial dimensions to estimate Height and Weight opening the doors for further advancements in health monitoring applications. Future work involves extending the dataset and incorporating deep learning for improved accuracy.

Table of Contents

Table of Figures

Table of Tables

Introduction:

The goal of this project is to predict height and weight using facial landmark-based features extracted from images. Using machine learning techniques, the relationship between facial dimensions and height and weight can be analyzed and modeled. This project uses a shape predictor model to calculate facial features and links these to BMI-related outcomes.

Literature Work:

Anthropometry and Facial Anthropometry

Anthropometry is the scientific study of human body measurements and proportions. It is widely used in fields like ergonomics, forensics, healthcare, and nutrition to analyze physical variations across populations, assess growth and development, and design equipment or environments that fit human dimensions. By measuring body parameters such as height, weight, and limb lengths, anthropometry provides a quantitative framework for understanding human body composition and its relationship to health and functionality.

Facial Anthropometry, a specialized subset of anthropometry, focuses on measuring the dimensions and proportions of facial features. It has applications in various domains, including forensic identification, craniofacial surgery, orthodontics, and biometric authentication. The shape and size of facial features, such as the eyes, nose, jaw, and mouth, are influenced by genetic, environmental, and nutritional factors, making them valuable indicators of individual identity and health status.

Relationship Between Facial Features and Height/Weight

Facial features, when analyzed quantitatively, can reflect underlying skeletal structure, genetic inheritance, and nutritional status. The study of relationships between specific facial landmarks and anthropometric parameters like height and weight is grounded in the following scientific logics:

Skeletal Correlation:

The distances between facial landmarks, such as the jaw width, nose-to-chin length, and eye distance, are directly influenced by the underlying craniofacial skeleton, which grows proportionally to overall body stature.

1. **Proportional Growth Patterns:**

Human growth follows specific patterns, and facial dimensions often scale in proportion to height and weight. For example, taller individuals tend to have longer facial measurements due to proportional growth.

2. **Body Fat Distribution:**

Features such as the width of the jaw or mouth can be indirectly influenced by weight, as soft tissue deposition around the face varies with body fat levels.

3. **Genetic Influence:**

Genetic factors link overall body proportions to facial structure, which allows for predictive modeling of height and weight using facial dimensions.

Requirements:

Dataset: Facial images annotated with specific measurements.
Libraries and Tools

- Python
- OpenCV
- dlib (for shape_predictor_68_face_landmarks)
- NumPy, Pandas, Matplotlib, Seaborn (for data analysis and visualization)
- Scikit-learn (for model development and evaluation)

Data Collection

About 200 facial images of athletes with know height and weight were collected from internet. The athletes include Footballers, Boxers, Basketball Players, Cricketers and more.

- Average Image size = 100kb
- Average Image size = 640 X 520
  Image Source = Pinterest
- Image Format = .jpeg & .jpg

Data Annotation:

The facial features were extracted from the images using dlib (shape_predictor_68_face_landmarks.dat).

shape_predictor_68_face_landmarks

The shape_predictor_68_face_landmarks.dat file is a crucial component in facial landmark detection, specifically designed to identify 68 distinct facial landmarks. This model is widely utilized in various applications, including face recognition, emotion detection, and facial feature analysis. The landmarks correspond to specific points on the face, such as the corners of the eyes, the tip of the nose, and the contour of the lips, allowing for precise facial feature mapping.

Key Features of the Model

Accuracy: The model is trained on a diverse dataset, ensuring high accuracy in landmark detection across different facial structures and expressions.

Robustness: It effectively handles variations in facial poses and expressions, making it suitable for real-time applications.

Open-source: Being part of the dlib library, it is freely available for use and modification, promoting innovation in facial recognition technologies.

Downloading the Model

To utilize the shape_predictor_68_face_landmarks.dat, you can download it directly from the official dlib repository. The download link is available at dlib.net. Ensure you have the necessary dependencies installed to integrate this model into your projects.

Implementation Example

Here's a simple example of how to use the shape_predictor_68_face_landmarks.dat in Python with dlib:

```
import dlib
import cv2

# Load the predictor
predictor_path = 'shape_predictor_68_face_landmarks.dat'
face_detector = dlib.get_frontal_face_detector()
shape_predictor = dlib.shape_predictor(predictor_path)

# Load an image
image = cv2.imread('face.jpg')
# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```python
# Detect faces
faces = face_detector(gray)
for face in faces:
    landmarks = shape_predictor(gray, face)
    for n in range(0, 68):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        cv2.circle(image, (x, y), 3, (255, 0, 0), -1)

cv2.imshow('Landmarks', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

This code snippet demonstrates how to load the model, detect faces in an image, and draw the landmarks on the detected faces.

(Restack.io/p, 2025)

Usage:

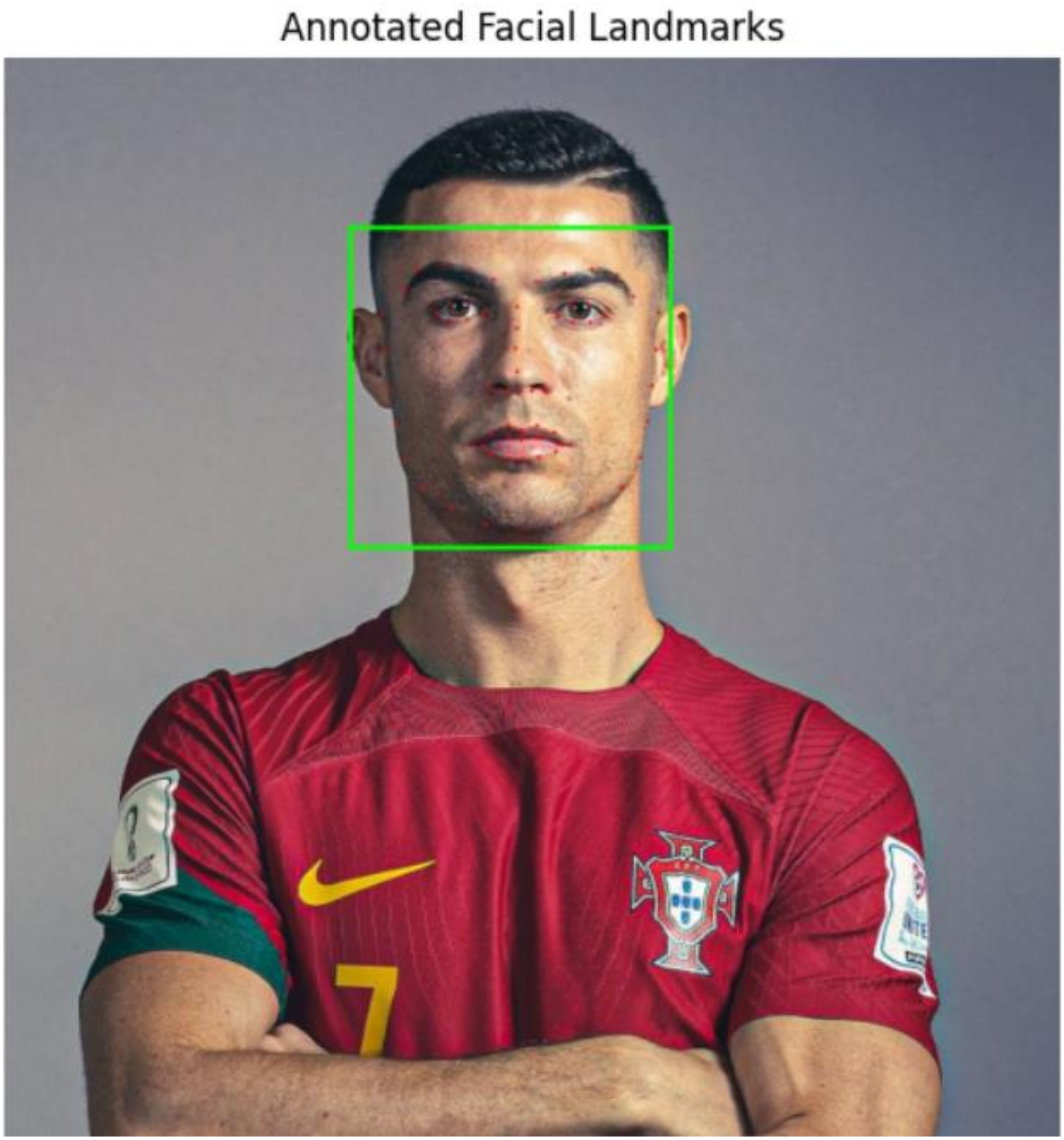

Figure 1Feature Extraction Result

```
# Helper function to extract distances between landmarks
def extract_features(shape):
    # Key landmarks
    left_eye = shape[36]
```

```
    right_eye = shape[45]
    nose_tip = shape[33]
    left_mouth = shape[48]
    right_mouth = shape[54]
    chin = shape[8]
    left_jaw = shape[0]
    right_jaw = shape[16]

    # Feature calculations (distances in pixel units)
    features = {
        "eye_distance": ((right_eye[0] - left_eye[0]) ** 2 + (right_eye[1] - left_eye[1]) ** 2) ** 0.5,
        "nose_to_chin": ((chin[0] - nose_tip[0]) ** 2 + (chin[1] - nose_tip[1]) ** 2) ** 0.5,
        "mouth_width": ((right_mouth[0] - left_mouth[0]) ** 2 + (right_mouth[1] - left_mouth[1]) ** 2) **
0.5,
        "jaw_width": ((right_jaw[0] - left_jaw[0]) ** 2 + (right_jaw[1] - left_jaw[1]) ** 2) ** 0.5,
        "nose_to_eye": ((nose_tip[0] - left_eye[0]) ** 2 + (nose_tip[1] - left_eye[1]) ** 2) ** 0.5,
    }
    return features
```

The above code is used to calculate the required facial features from extracted facial landmarks using the dlib predictor model.

The extracted features were stored in a csv file along with the image name.

eye_distance	nose_to_chin	mouth_width	jaw_width	nose_to_eye	image_name
200.2023976	160.7015868	148.5698489	328.2575209	147.1054044	image185.jpg
86.57944329	88.36288814	73.68174808	151.6047493	67.95586803	image184.jpg
110.2225022	99.18165153	74.54528825	183.9836949	81.27115109	image183.jpg
85	70.06425622	52.0096145	131.0610545	55.1724569	image182.jpg
126.0357092	113.039816	91.08786966	211.0379113	80.28075735	image180.jpg
114.0394669	96.13012015	69.02897942	165.0757402	72.83543094	image177.jpg
70.11419257	55.03635162	45.09988914	109.2931837	46.40043103	image178.jpg
95.08417324	85.0940656	70.11419257	165.0757402	71.70076708	image179.jpg
159.0503065	136.2350909	94.08506789	247.0020243	105.4751155	image181.jpg
93.00537619	76	62.00806399	135.0037037	67.00746227	image176.jpg
70.17834424	63.19810124	50.15974482	120.3370267	46.22769733	image165.jpg
97.08243919	94.1328848	74.06078585	164.1097194	65.29931087	image167.jpg
36	35.0142828	28	62.03224968	24.8394847	image168.jpg
120.0374941	103.0194156	95.25754563	192.0104164	98.71676656	image169.jpg
86.092973	73.10950691	47.01063709	143.0873859	60.87692502	image175.jpg
75.02666193	62.07253821	60.03332408	124.2577965	53.90732789	image173.jpg
100.0449899	97.0051545	69.007246	163.0766691	69.42621983	image171.jpg
71.02816343	63.07138812	46.04345773	116.0172401	49.57822102	image170.jpg
85.0529247	70.0285656	49.01020302	146.0547842	59.41380311	image172.jpg
121.4989712	109.1650127	70.06425622	204.1984329	83.48652586	image174.jpg
92.08691547	71.11258679	54.00925847	155.0806242	65.29931087	image166.jpg
81.0061726	68.11754546	61	131.0038167	62.00806399	image163.jpg
126.015872	115.0043477	64.07027392	205.0097559	87.20665112	image160.jpg
78.31347266	72.11102551	50.15974482	140.0892573	46.61544808	image157.jpg
73.10950691	56.08029957	45.09988914	124.2577965	54.45181356	image162.jpg
52	49.01020302	34	89.0056178	38.18376618	image158.jpg
91.02197537	92	52	148.0033783	71.21797526	image161.jpg
159.2011306	132.0340865	101.0445446	272.413289	131.605471	image159.jpg
123.0650235	107.0046728	71.0070419	191.1282292	83.73768566	image156.jpg

features_dataset

Figure 2Facial Features

Data Labeling:

The output labels (height and weight) were added next to the features dataset using a python script. The Height and Weight was collected from internet. We stored the output labels next to the image name in .txt file then used python script to add that in the features_dataset. The data was also sorted by image name in the script.

	A	B	C	D	E	F	G	H
1	eye_distance	nose_to_chin	mouth_width	jaw_width	nose_to_eye	image_name	height	weight
2	109.2245394	100.3194896	88.20430828	190.9476368	82.73451517	image1.jpeg	173	67
3	116.0172401	93.13431162	69	192.0104164	81.32035416	image2.jpeg	188	83
4	89.14033879	90.19977827	48.16637832	161.697248	62.28964601	image3.jpeg	176	66
5	145.0034482	133	82	260.1922366	113.8463877	image4.jpeg	185	81
6	154.3534904	130.0499904	105.1189802	253.2133488	94.87360012	image5.jpeg	155	74
7	109.1650127	88.20430828	72.44308111	184.4586675	82.13403679	image6.jpeg	195	87
8	102.019606	87.09190548	63.12685641	175.0114282	75.66372975	image7.jpeg	155	68
9	133.2403843	130.1883251	73.33484847	227.879354	104.6518036	image8.jpg	188	85
10	110.0181803	106.0424443	69.0651866	187.0026738	75.69015788	image9.jpg	185	81
11	112.0401714	110.0727032	67.11929678	201.1218536	88.23831367	image10.jpeg	182	82
12	147.0034013	126.1942946	112.1115516	245.0510151	100.4987562	image11.jpg	179	73
13	129.1394595	124.0644994	96.13012015	208.0865205	87.7268488	image12.jpg	198	92
14	172.0726591	153.0522787	106.117859	296.1367927	114.5512986	image13.jpeg	195	92
15	101.0445446	81.15417426	71.1758386	174.0114939	75.74298647	image14.jpg	155	71
16	121.1032617	97.02061637	82.00609733	197.040605	85.14693183	image15.jpg	182	76
17	77	66.00757532	56.0357029	125.0039999	56.08029957	image16.jpg	155	66
18	102.0049018	88	72.11102551	177	82.75868534	image17.jpg	198	89
19	87.69264507	73.68174808	65.76473219	138.9244399	55.00909016	image18.jpg	155	76
20	102.5914226	93.13431162	59.41380311	172.5398505	77.38862966	image19.jpg	176	64
21	110.0181803	99.18165153	71.02816343	165.1938256	65.60487787	image20.jpg	192	83
22	133.0601368	115.277925	92.19544457	218.1857007	94.17536833	image21.jpg	179	68
23	136.0918807	124.0644994	117.0170928	209.038274	93.34880824	image22.jpg	179	73
24	132.1362933	108	87.82368701	211.1516043	98.41239759	image23.jpg	185	83
25	101.4938422	103.3102125	68.11754546	167.9672587	68.00735254	image24.jpg	179	73
26	51.0881591	47.09564736	35	85.14693183	34.6554469	image25.jpg	155	69
27	65.00769185	56	33	101.0445446	45.27692569	image26.jpg	179	72
28	59.07622195	49.16299421	40.04996879	95.25754563	49.81967483	image27.jpg	178	75
29	103.5857133	85.0529247	62.51399843	170.8478856	64.07027392	image28.jpg	169	62
30	79.00632886	63.00793601	55.03635162	128.0039062	51.6139516	image29.jpg	181	79

processed_dataset

Figure 3Labeled Facial Features

Data Processing:

Feature values were scaled to ensure uniformity and to improve model performance.

The minimum and maximum values of features are as follows:

- eye_distance: Min = 36.0, Max = 226.035
- nose_to_chin: Min = 31.40, Max = 218.146
- mouth_width: Min = 24.33, Max = 158.028
- jaw_width: Min = 62.03, Max = 397.283
- nose_to_eye: Min = 23.70, Max = 167.621

Scaling was done to normalize the feature ranges and ensure that all features contribute equally to the model.

Scaling Parameters:


```
{
  "feature_names": [
    "eye_distance",
    "nose_to_chin",
    "mouth_width",
    "jaw_width",
    "nose_to_eye"
  ],
  "min_values": [
    36.0,
    31.400636936215164,
    24.33105012119288,
    62.03224967708329,
    23.706539182259394
  ],
  "max_values": [
    226.035395458322,
    218.14673960433143,
```

158.02847844613325,

397.2832742515093,

167.621597653763

]

}

Scaled Dataset was stored in scaled_dataset.csv

eye_distance	nose_to_chin	mouth_width	jaw_width	nose_to_eye	image_name	height	weight
0.38532053037122016	0.36905108974693046	0.4777448524168192	0.3845339094425178	0.410158441	image1.jpeg	173	67
0.421064928	0.3305754379704397	0.33410478001299315	0.3877040103666359	0.40033208189118397	image2.jpeg	188	83
0.2796339001168906	0.3148614107279104	0.17827813513395738	0.2972846941068101	0.26809638433333327	image3.jpeg	176	66
0.5735955029428571	0.5440507813132059	0.43133926060752326	0.5910794372653103	0.6263406311246891	image4.jpeg	185	81
0.6227970853476519	0.5282538807645991	0.6042594169470636	0.5702625349015755	0.4945073968979904	image5.jpeg	155	74
0.38500729052953353	0.3041759401508671	0.3598575648513912	0.36517835533923965	0.40598599084635767	image6.jpeg	195	87
0.3474068912248194	0.29821917430839745	0.2901761595059532	0.33699875687064557	0.36102678289316303	image7.jpeg	155	68
0.5116961713048908	0.5289946445027334	0.36652760611747837	0.49469529460355793	0.5624516662332282	image8.jpg	188	85
0.38949680998787406	0.39969673438853026	0.33459234809971505	0.3727667178921609	0.36121041988596936	image9.jpg	185	81
0.40013688599350783	0.4212782231398764	0.32003791840343576	0.4148819652696583	0.4484018224148626	image10.jpeg	182	82
0.5841196112623076	0.5076071540485694	0.6565608820525186	0.5459156035454362	0.5335940369586483	image11.jpg	179	73
0.4901163768757615	0.4962023897367911	0.5370265601072562	0.4356564487018911	0.4448478866321165	image12.jpg	198	92
0.7160384976888534	0.6514280082734731	0.611730606	0.6982962791415546	0.6312387343951873	image13.jpeg	195	92
0.3422759453610231	0.26642343060460205	0.3503791289242436	0.3340161132641278	0.3615775016485899	image14.jpg	155	71
0.44782847699482053	0.3513860717415105	0.43138486607922166	0.4027082555427807	0.42692122214256223	image15.jpg	182	76
0.21574928134370633	0.18531545179352427	0.23713734198566053	0.18783462433517178	0.22495047240237687	image16.jpg	155	66
0.347329515558307	0.3030818970523459	0.3573739299753895	0.34293034739821887	0.41032638828573853	image17.jpg	198	89
0.27201587866778265	0.22640960394274512	0.3099063503894448	0.22935706256236849	0.2175071275243865	image18.jpg	155	76
0.35041589217004887	0.3305754379704397	0.2624040972844537	0.32962643718267143	0.37301232437850507	image19.jpg	176	64
0.38949680998787406	0.36295812134373673	0.34927458136017964	0.307714424	0.2911324160943432	image20.jpg	192	83
0.5107476770116568	0.44915147839108627	0.5075968573366818	0.46578068253235544	0.4896557031327615	image21.jpg	179	68
0.5267012520814449	0.4962023897367911	0.6932522473212932	0.43849537676117806	0.48391231465503726	image22.jpg	179	73
0.5058862484086853	0.4101791789460615	0.47489796688026253	0.4447991018546457	0.5190968839825697	image23.jpg	185	83
0.34464022883623247	0.385066004	0.3275043947449261	0.31598712985800054	0.30782611515382013	image24.jpg	179	73
0.079396573	0.084044648	0.079799215	0.068947387	0.076078958	image25.jpg	155	69
0.15264362611302015	0.13172624602240096	0.064840065	0.11636741455150718	0.1498827623566495	image26.jpg	179	72
0.12143117809875967	0.095115009	0.11757083785938785	0.099105725	0.1814482509572536	image27.jpg	178	75
0.35564802618810654	0.28730070934328916	0.285592242	0.32457957740801674	0.2804691542705961	image28.jpg	169	62
0.2263069401186725	0.16925279092999623	0.22966261869636348	0.19678286322035715	0.19391586062220356	image29.jpg	181	79

scaled_dataset

Figure 4Scaled Dataset

Model Training:

The scaled dataset was split into training and testing sets with an 80-20 split ratio using train_test_split from Scikit-learn.

Separate Random Forest models were created and trained for height and weight predictions.

Hyperparameters included:

- Number of estimators: 100
- Random state: 42 (for reproducibility)

The training process was monitored to minimize error metrics and optimize performance.

Random Forest Model:

The Random Forest model is an ensemble learning method primarily used for regression and classification tasks. It operates by building multiple decision trees during training and combining their outputs to improve prediction accuracy and control overfitting. The model works by averaging the predictions of individual trees (for regression) or taking a majority vote (for classification).

Number of Estimators:

This parameter specifies the number of decision trees in the forest.

In this project, the number of estimators was set to 100, meaning 100 decision trees were built and their results aggregated for the final prediction.

A larger number of estimators generally improves performance but increases computation time.

Random State:

The random state is a seed value used for reproducibility. It ensures that the random sampling of the dataset (e.g., during splitting or tree creation) produces the same results every time the code is run.

In this project, the random state was set to 42, a commonly used value for consistency in experiments.

.Joblib files were saved for trained Height and Weight models separately.

Work Flow of the Training

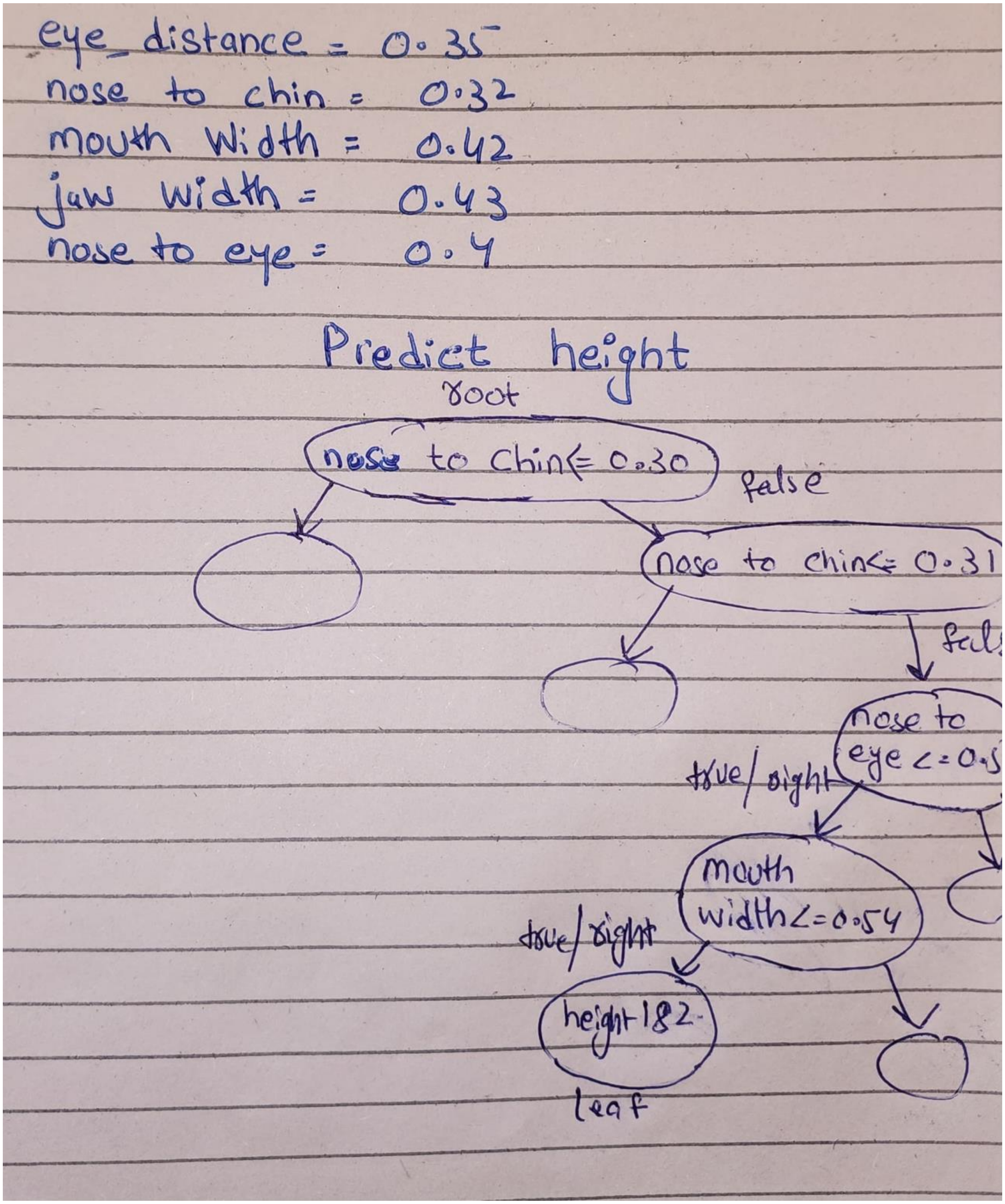

Figure 5Forest Regressor

Example Decision Tree Table for Height Prediction

Node Level	Condition	Decision (Left/Right)	Next Node	Predicted Value (if Leaf)
Root	nose_to_cl <= 0.30	True (Left)	Leaf	155
		False (Right)	Node 2	
Node 2	nose_to_cl <= 0.31	True (Left)	Leaf	155
		False (Right)	Node 3	
Node 3	nose_to_ej <= 0.55	True (Left)	Node 4	
		False (Right)	Leaf	188
Node 4	mouth_width <= 0.54	True (Left)	Node 5	
		False (Right)	Leaf	198
Node 5	jaw_width <= 0.43	True (Left)	Leaf	182
		False (Right)	Leaf	185

Table 1Decision Tree Height

Model Evaluation:

Test dataset was used to evaluate the models.

Following are the results:

Height Model Performance	Results
Mean Absolute Error (MAE)	9.95
Mean Squared Error (MSE)	145.03
Root Mean Squared Error (RMSE)	12.04
Root Mean Squared Log Error (RMSLE)	0.07
R^2 Score	-0.39
Adjusted R^2 Score	-0.59

Table 2Evaluation Matrix of Trained Height Model

Mean Absolute Error: The average absolute difference between the predicted and actual height is approximately 9.95 units.

Mean Squared Error (MSE): The squared errors are larger, indicating that the model's errors can sometimes be significantly high.

Root Mean Squared Error (RMSE): This value represents the error in the same units as the target variable (height). It shows that the model's predictions deviate by about 12 units on average.

Root Mean Squared Log Error (RMSLE): RMSLE indicates that the relative differences between the log-transformed predictions and actual values are small.

R^2 Score: A negative R^2 score suggests that the model performs worse than a simple mean-based prediction. This indicates the model struggles to capture the variance in the data.

Adjusted R^2 Score: Adjusted R^2 also being negative further emphasizes the need for better feature selection or a larger dataset.

Weight Model Performance	**Results**
Mean Absolute Error (MAE)	6.47
Mean Squared Error (MSE)	56.27
Root Mean Squared Error (RMSE)	7.50
Root Mean Squared Log Error (RMSLE)	0.10
R^2 Score	-0.10
Adjusted R^2 Score	-0.27

Table 3Evaluation Matrix of trained Weight Model

Mean Absolute Error (MAE): The average error in predicting weight is about 6.47 units.

Mean Squared Error (MSE): Some squared errors are significant, reflecting occasional large prediction deviations.

Root Mean Squared Error (RMSE): On average, the weight predictions deviate by about 7.50 units.

Root Mean Squared Log Error (RMSLE): RMSLE suggests a reasonable alignment between predicted and actual values when logarithmically transformed.

R^2 Score: A negative R^2 indicates the model explains less variance than a mean-based approach.

Adjusted R^2 Score: This reinforces that the model lacks sufficient explanatory power for the features provided.

Observation:

The results were satisfactory according to the size of the dataset. 200 images are not enough but the model did well on such a small dataset.

Model Working:

The pipeline follows

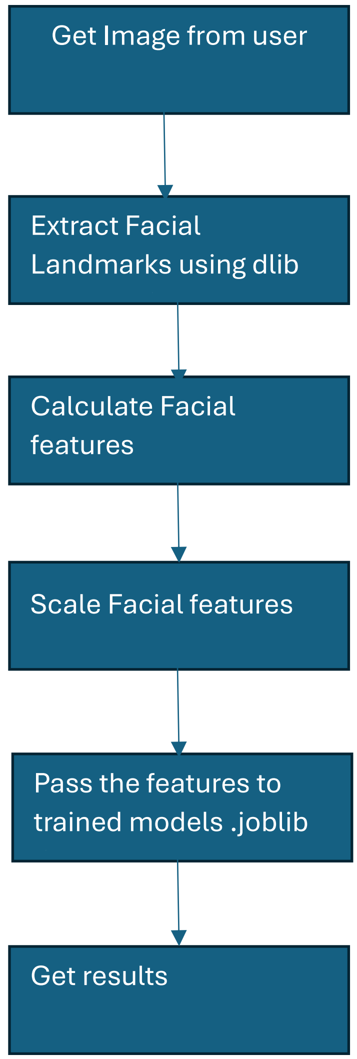

Figure 6Working of Trained Model

Future Work

- Extend the dataset to include more diverse demographics.
- Incorporate advanced deep learning techniques for feature extraction and prediction.
- Explore additional facial features to improve model accuracy.

Conclusion

This project demonstrates the feasibility of predicting BMI-related metrics using facial landmarks. While the results are promising, further refinements in data collection and model development are needed to enhance accuracy and applicability in real-world scenarios.