Child height Prediction based on parents height

Group Members:
Kaki.Himanth
Lovely professional University
12102363
Jayanth Reddy.Udumula
Lovely professional university
12103070

Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty, **Prof. Arnab Chakraborty** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Kaki.Himanth

Jayanth Reddy.Udumula

Project Objective

In this project we have a shortened ‘Predict Child height based on parents height’ Dataset from Kaggle. In this dataset, the target attribute is the Height . So, in this project we need to classification based on the attributes present in our dataset and predict whether a Child Height.

Our methodology for solving the problems in the given project is described below:

- Load the required dataset.
- Study the dataset.
- Describe the dataset.
- Visualise the dataset.
- Find out if the dataset needs to be pre-processed.
    - It will be determined on the basis of whether the dataset has null values or outliers or any such discrepancy that might affect the output of the models tobe trained.
- If the dataset is required to be pre-processed, take the necessary steps to pre-processthe data.
- Find out the principal attributes for training.
- Split the given dataset for training and testing purpose.
- Fit the previously split train data in the aforementioned 4 models.
- Calculate the accuracy of the 4 models and find out the classification reports.
- Plot the necessary graphs.
- Use each trained model to predict the outcomes of the given test dataset.

Project Scope

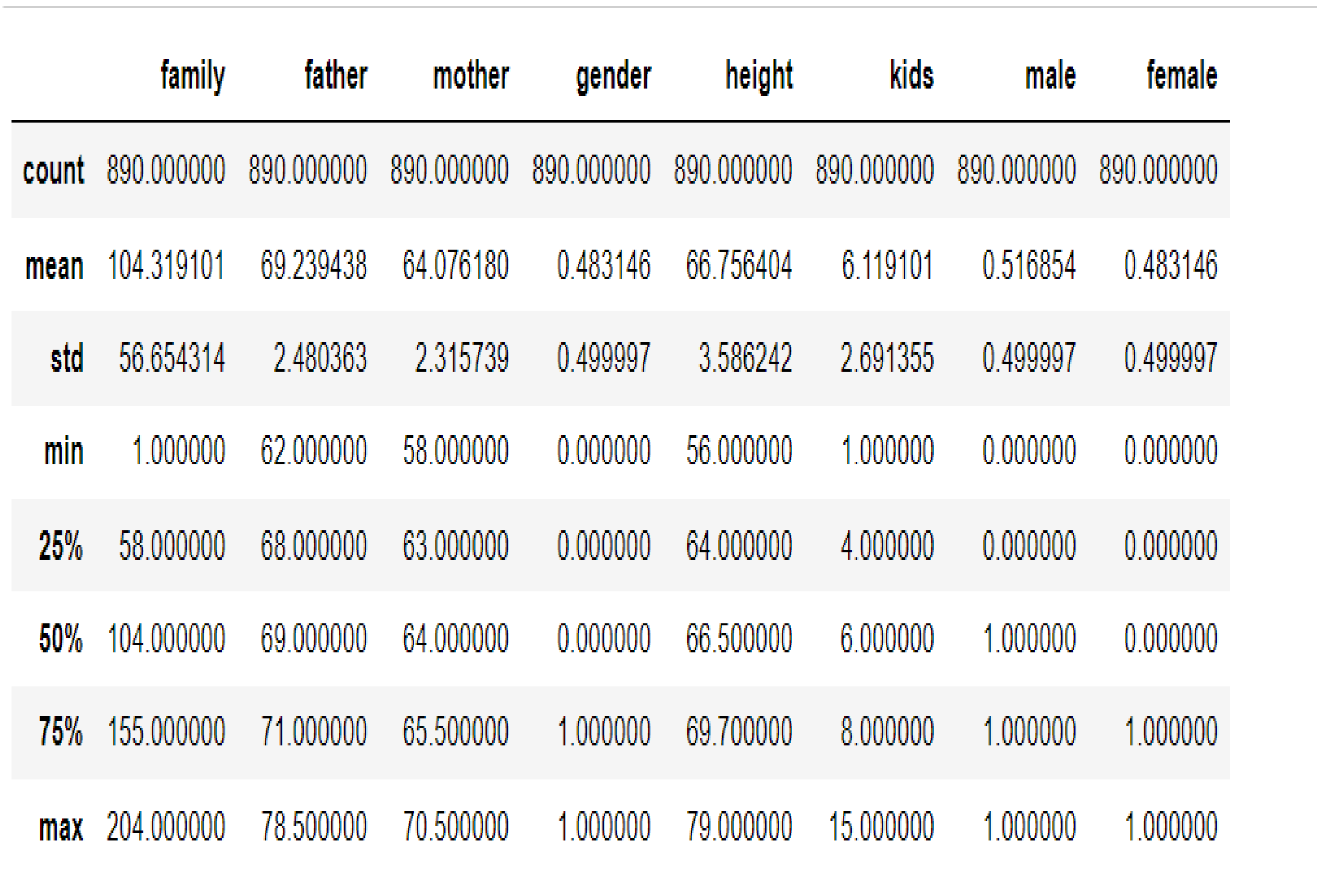
The scope of a child height prediction project based on parents' heights includes:

- Collecting accurate and diverse data on parents' heights and their children's heights.
- Analyzing the data and developing prediction models using statistical techniques and machine learning algorithms.
- Designing and implementing algorithms specific to predicting child height based on parental height.
- Assessing the accuracy and reliability of the prediction models through validation and comparison with actual measurements.
- Interpreting and effectively communicating the predicted heights to users, considering potential limitations and uncertainties.

Data Description

Source of the data: Kaggle. The given dataset is a shortened version of the original dataset in Kaggle.

The following table shows the statistic of the given dataset:-



	family	father	mother	gender	height	kids	male	female
count	890.000000	890.000000	890.000000	890.000000	890.000000	890.000000	890.000000	890.000000
mean	104.319101	69.239438	64.076180	0.483146	66.756404	6.119101	0.516854	0.483146
std	56.654314	2.480363	2.315739	0.499997	3.586242	2.691355	0.499997	0.499997
min	1.000000	62.000000	58.000000	0.000000	56.000000	1.000000	0.000000	0.000000
25%	58.000000	68.000000	63.000000	0.000000	64.000000	4.000000	0.000000	0.000000
50%	104.000000	69.000000	64.000000	0.000000	66.500000	6.000000	1.000000	0.000000
75%	155.000000	71.000000	65.500000	1.000000	69.700000	8.000000	1.000000	1.000000
max	204.000000	78.500000	70.500000	1.000000	79.000000	15.000000	1.000000	1.000000

Table: Number statistics of the given dataset

Now we will pre-process the data. The methodology followed is given below:

- Checking for null values.
  - ➢ If null values are present, we will fill them or drop the row containing the null value based on the dataset.
- Checking for outliers.
  - ➢ If outliers are present, they will either be removed or replaced by following a suitable method depending on the dataset.

Data Pre-processing

As the given dataset had Categorical and Non-categorical data mixed, we converted the categorical data into non-categorical data accordingly. We converted the binary categories into 0 and 1. We converted the other categorical attributes into suitable numerical values. The following table shows the conversion record:

Gender	M	0
	F	1

Table:Categorical to Numerical change in values.

We searched for null values in our dataset and formed the following table:

Column Name	Count of Null Values
Family	0
Father	0
Mother	0
Gender	0
Height	0
Kids	0
Male	0
Female	0

Table: Count of Null Values

To visualise the null values, we made a heatmap plot using seaborn library function heatmap
The heatmap plot is given below:

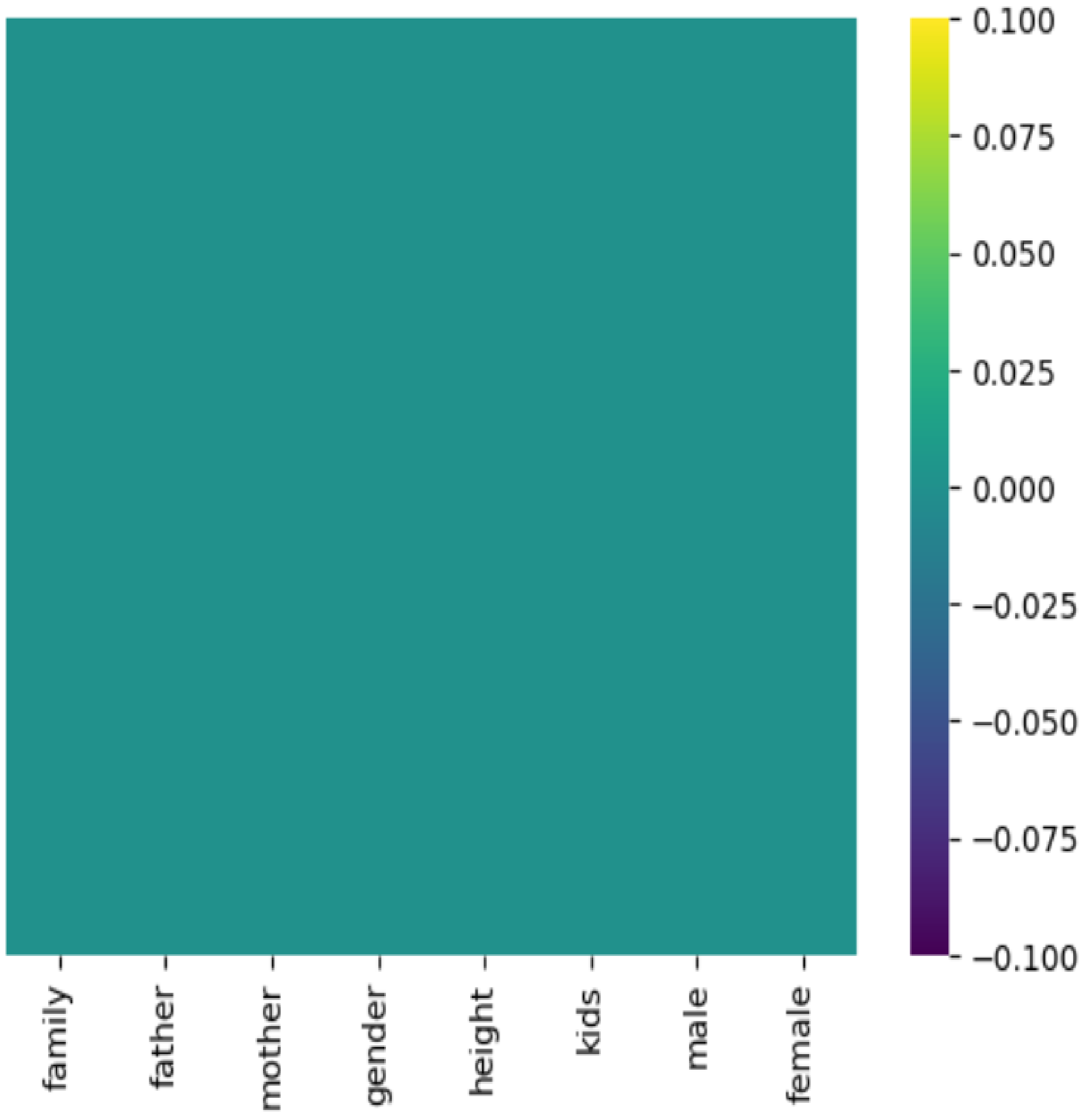
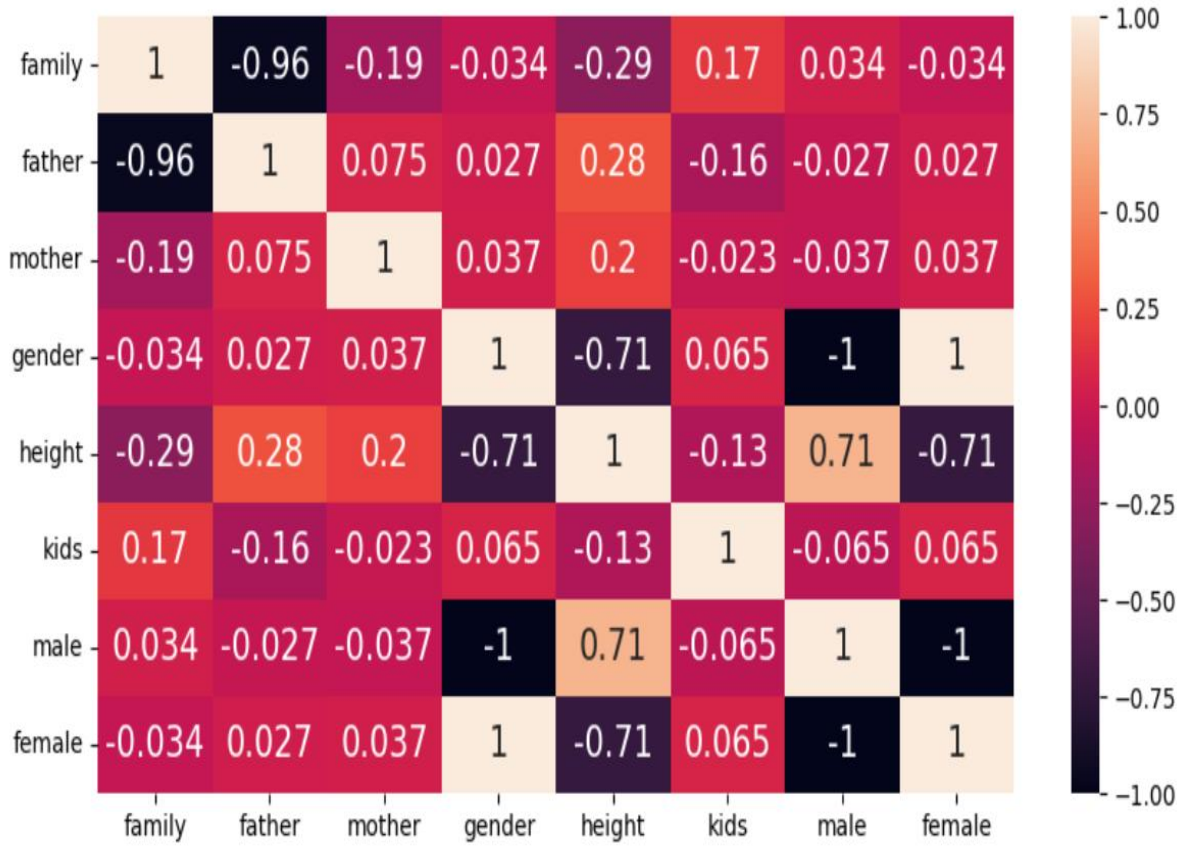


Fig: Heatmap of the given Data.

So, we are moving on to find if there are any outliers in our data and find the correlations of different attributes to our target i.e. 'Height' column in the dataset. The following table gives the correlation value of each attribute with our target attribute i.e. 'height':

Columns	Correlation Value
Father	0. 2768008110348267
Mother	0. 20203872284119695
Gender	-0. 7126729743888253
Height	1.0
Kids	-0.12847496133255157
Male	0.7126729743888252

Table : Correlation Values with Target Attribute.

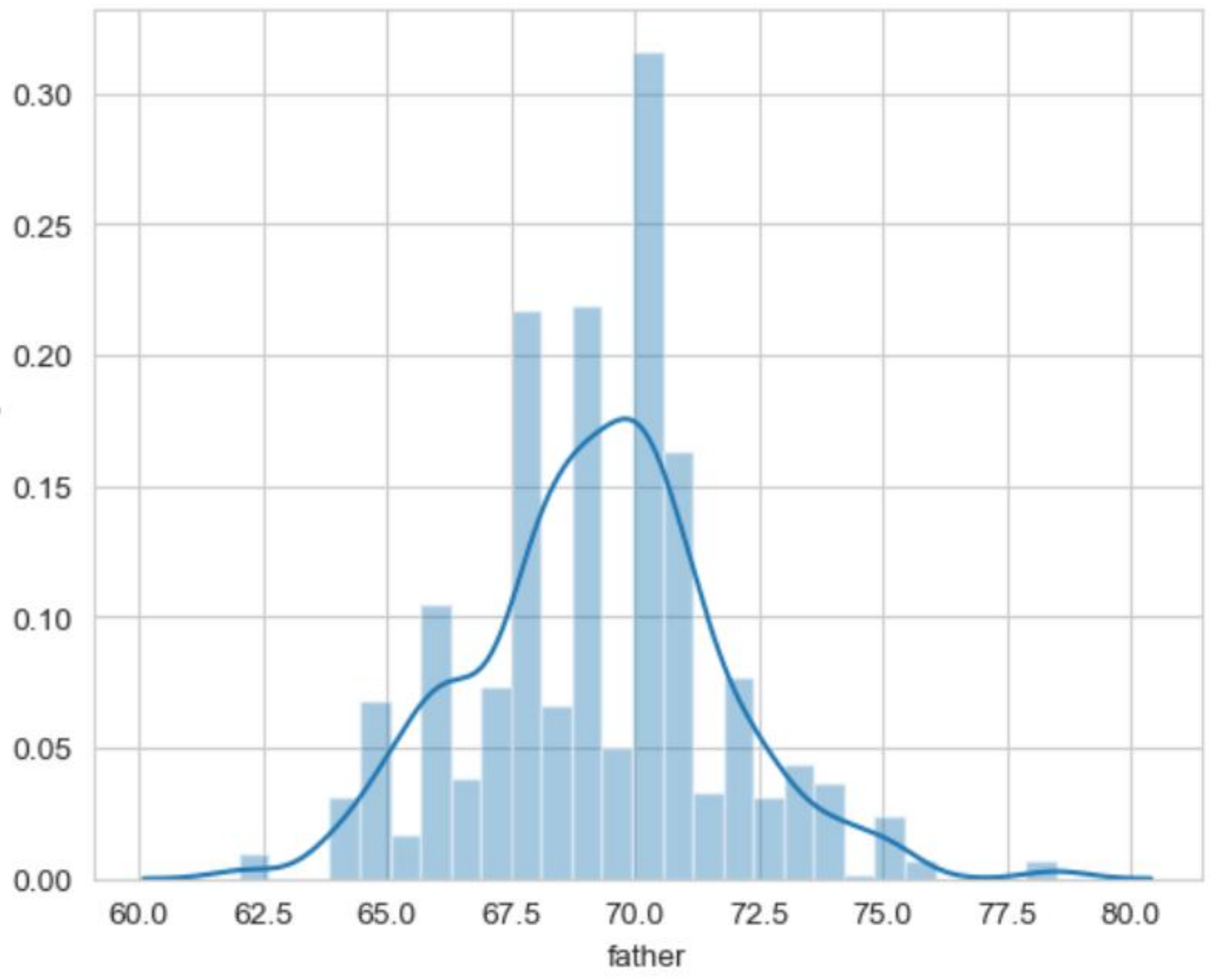


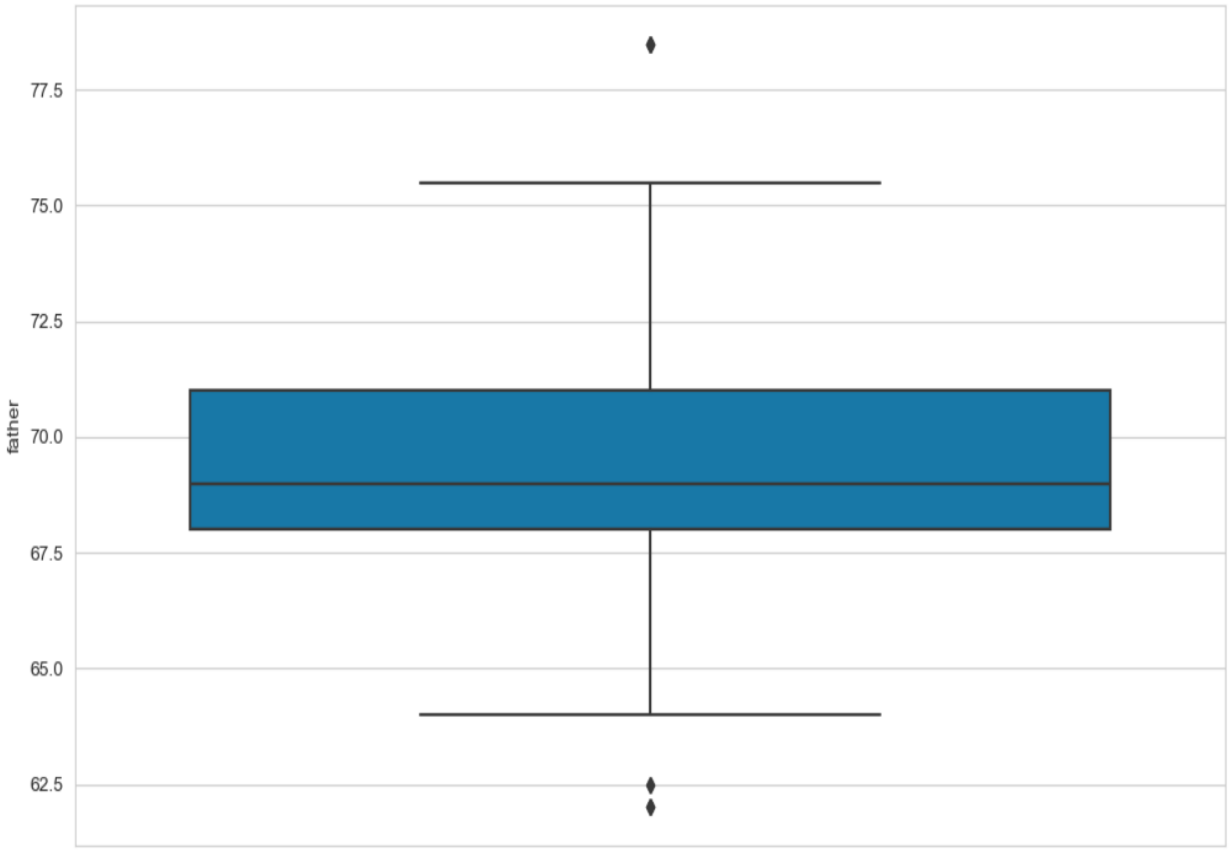
Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.

Most common causes of outliers on a data set:

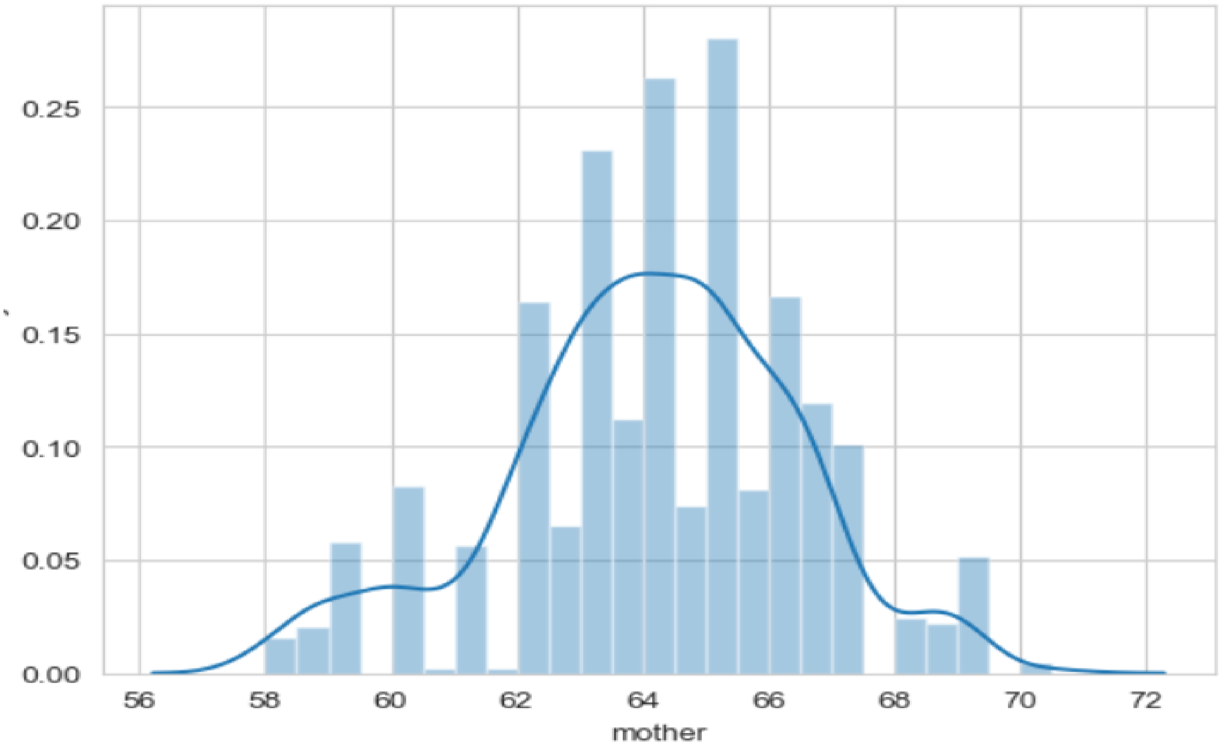
- Data entry errors (human errors)
- Measurement errors (instrument errors)
- Experimental errors (data extraction or experiment planning/executing errors)
- Intentional (dummy outliers made to test detection methods)
- Data processing errors (data manipulation or data set unintended mutations)
- Sampling errors (extracting or mixing data from wrong or various sources)
- Natural (not an error, novelties in data)

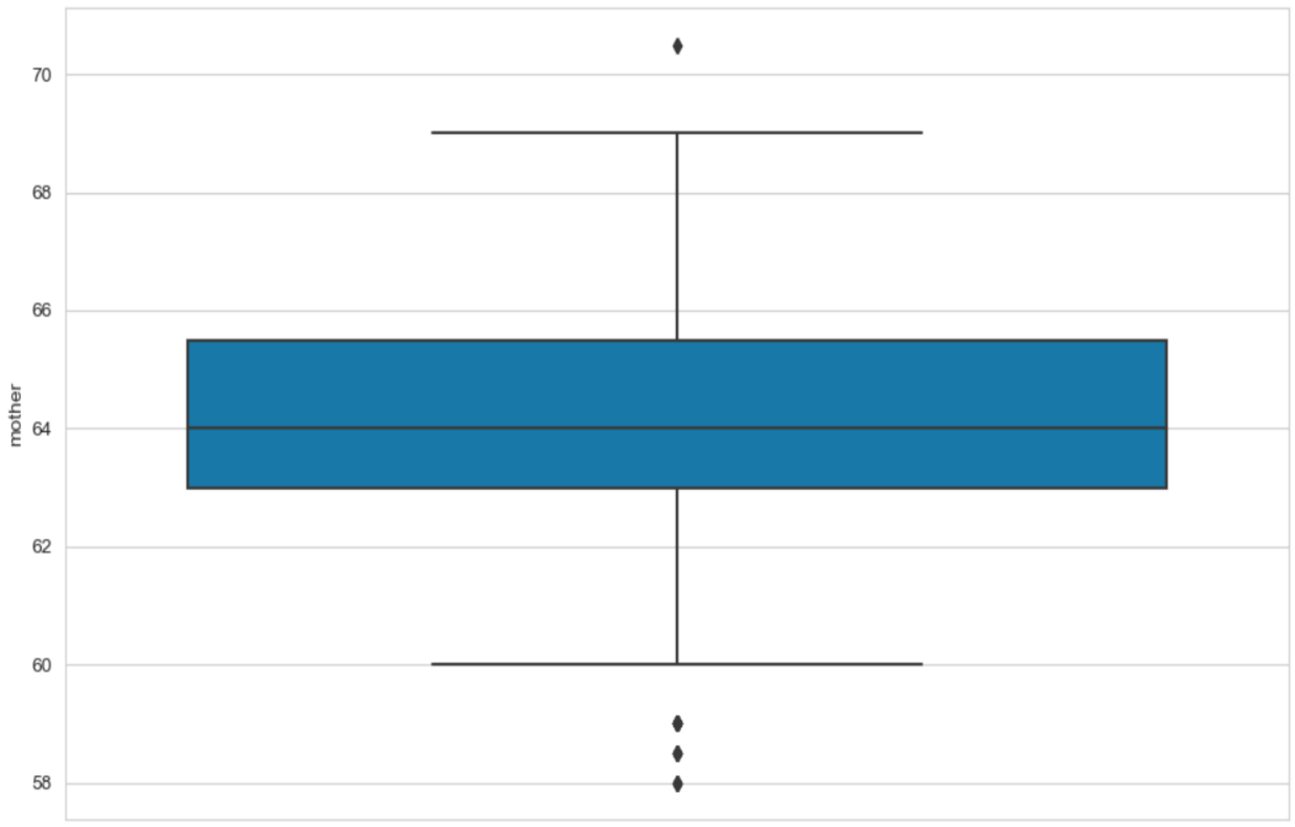
We plot distribution graph and boxplot to visualise the outliers. The plots are given below:



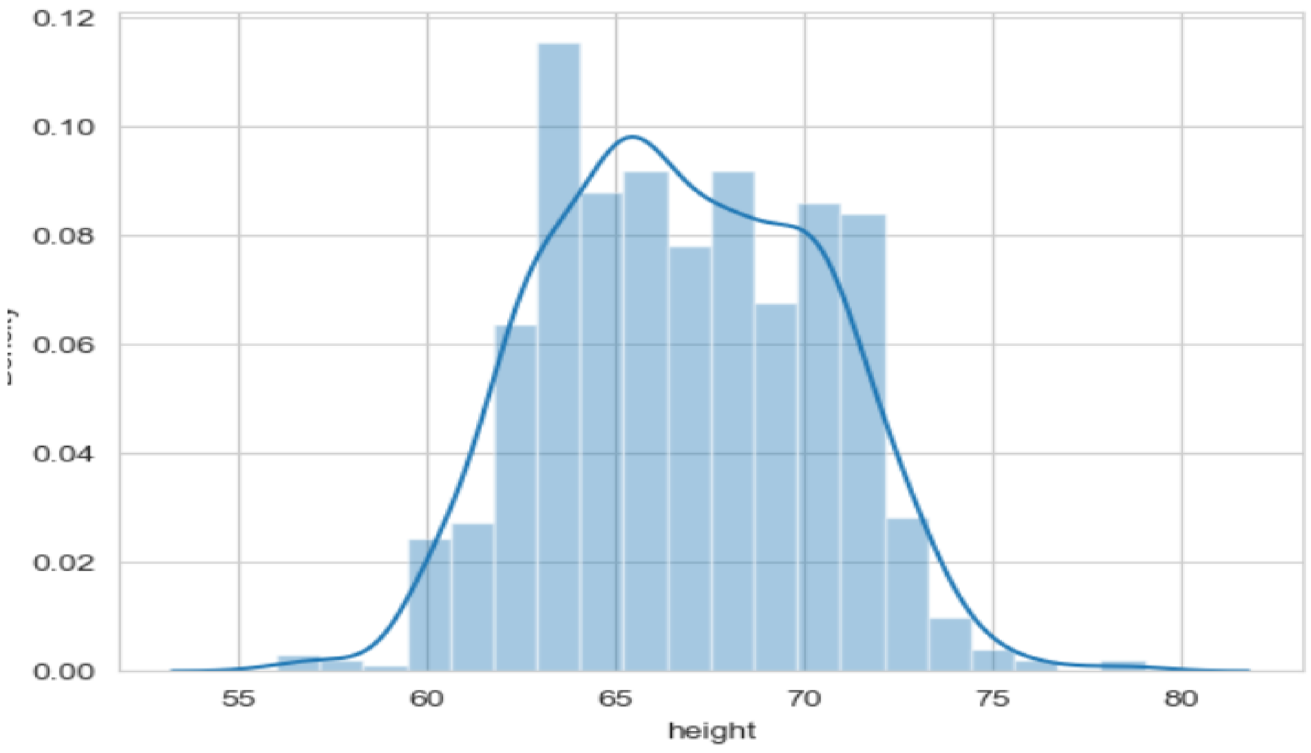


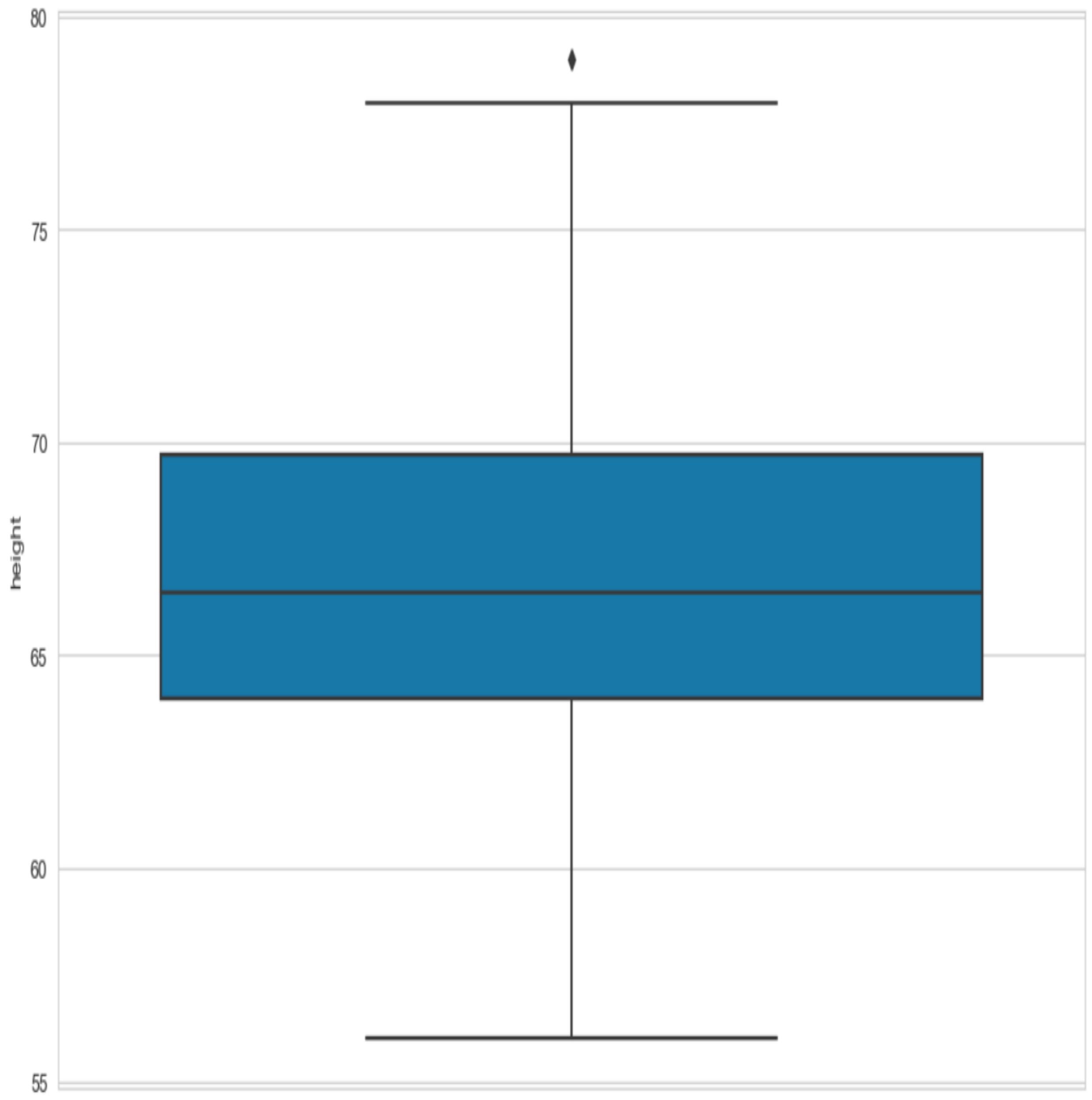
Distribution plot and boxplot of Father Height.



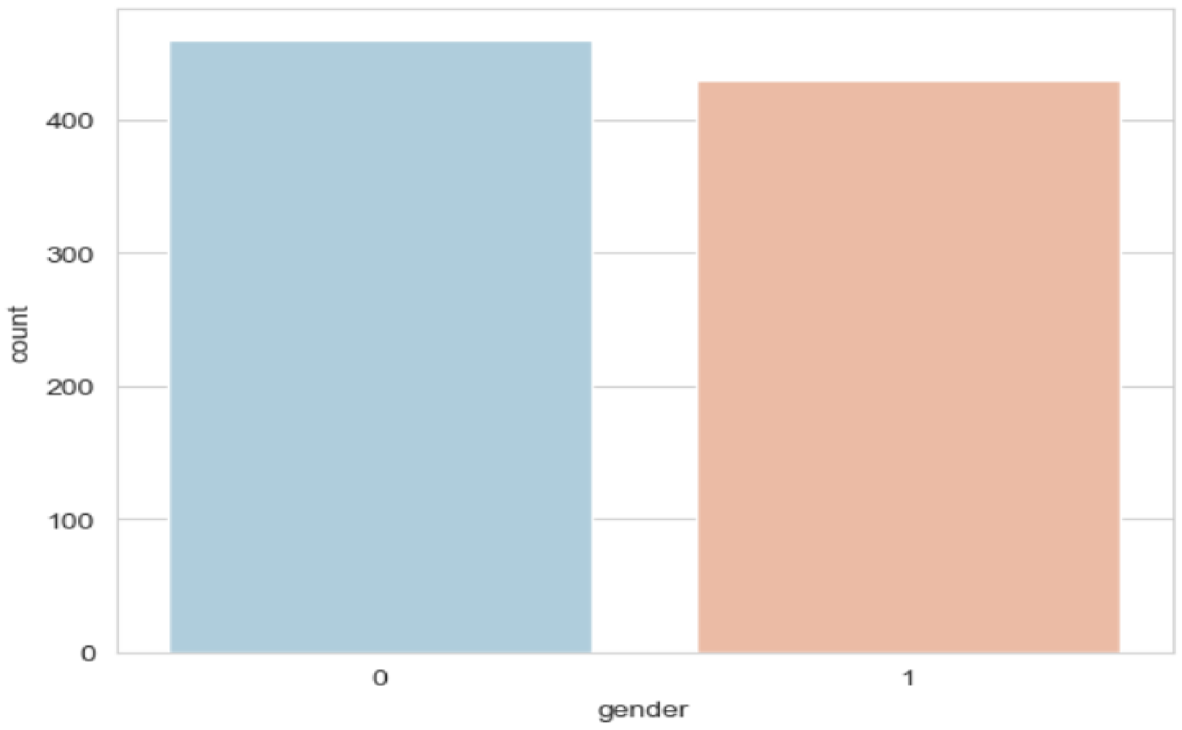


Distribution plot and boxplot of Mother Height.

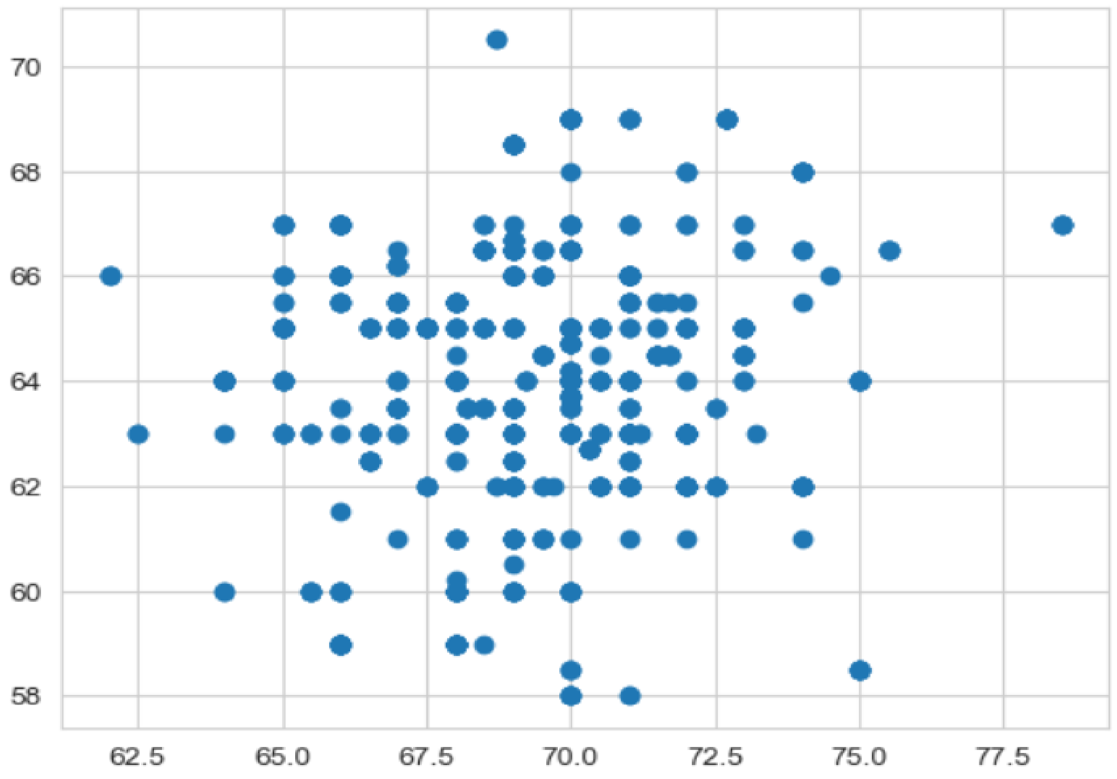




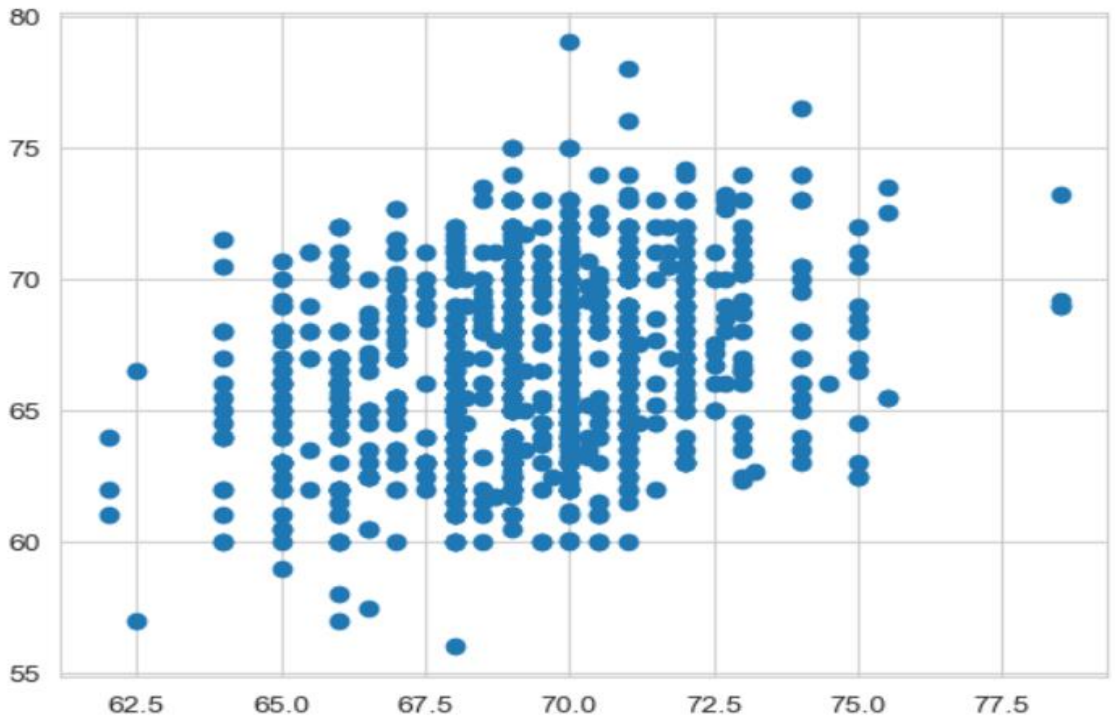
Distribution plot and boxplot of Child Height.



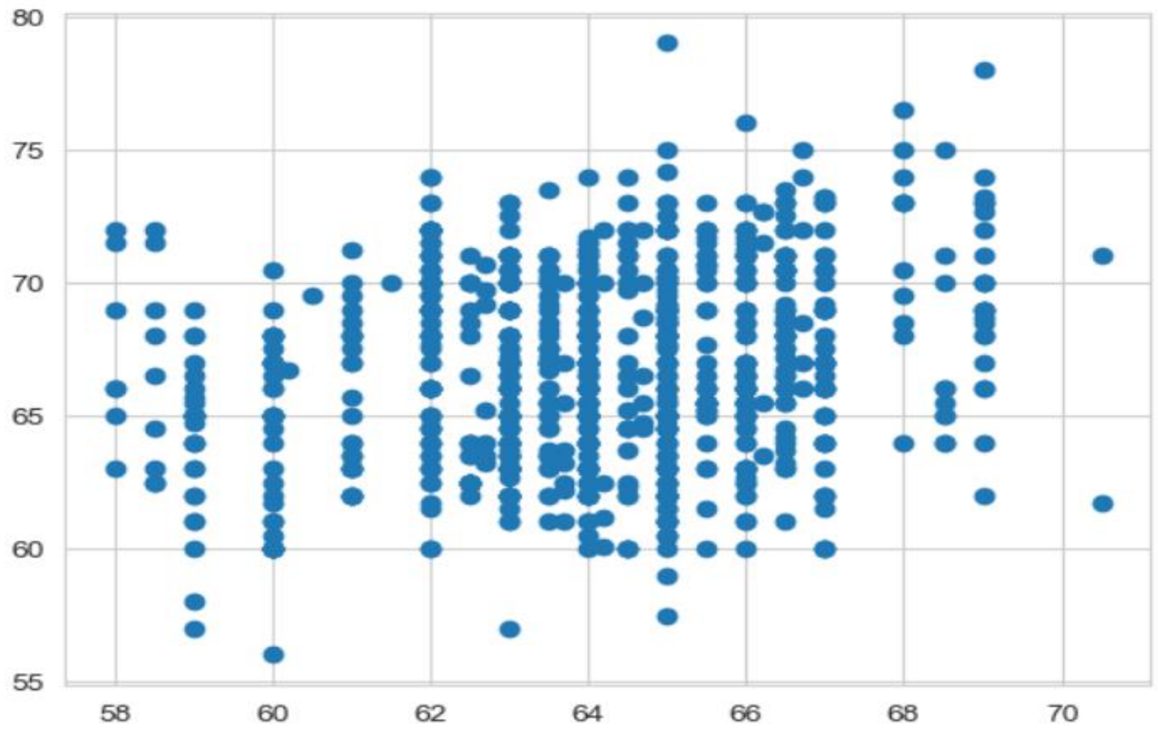
Bargraph of Gender.



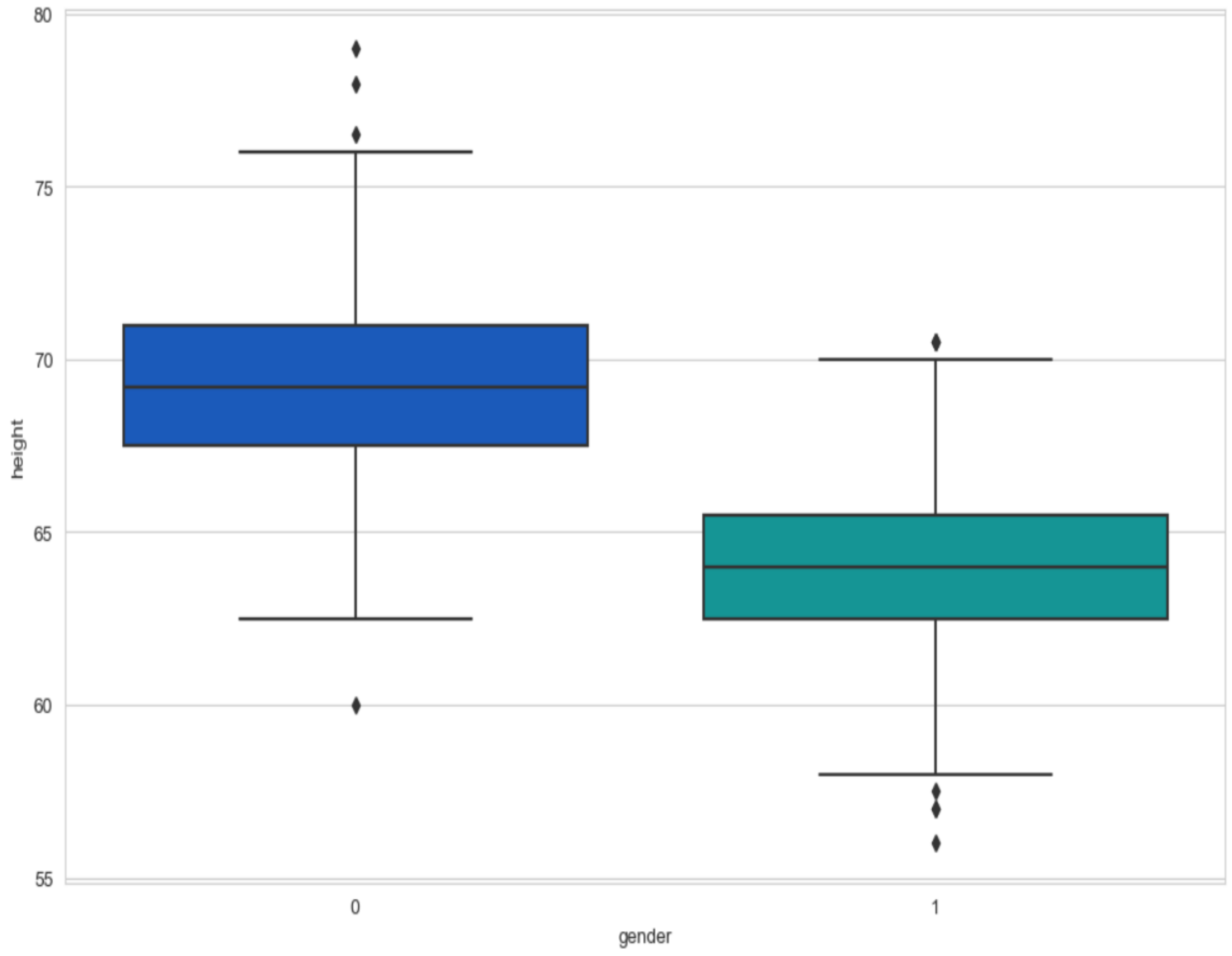
Scatter plot of Father and mother height.



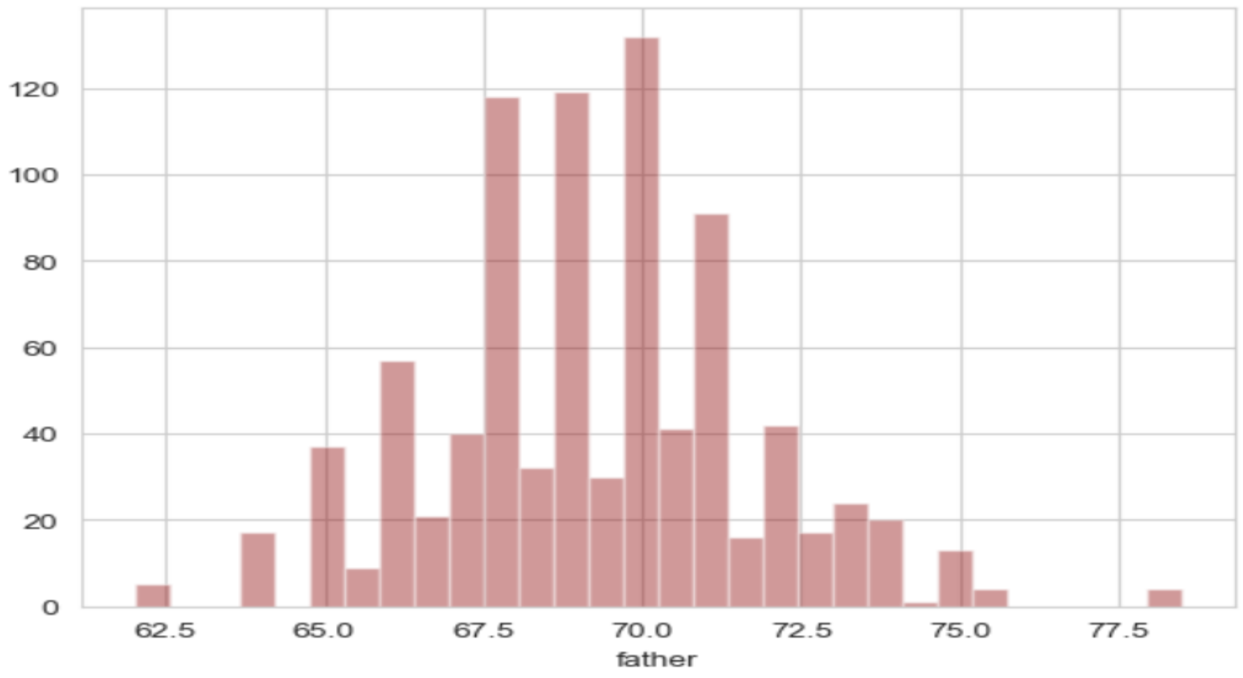
Scatter plot of Father and child height.

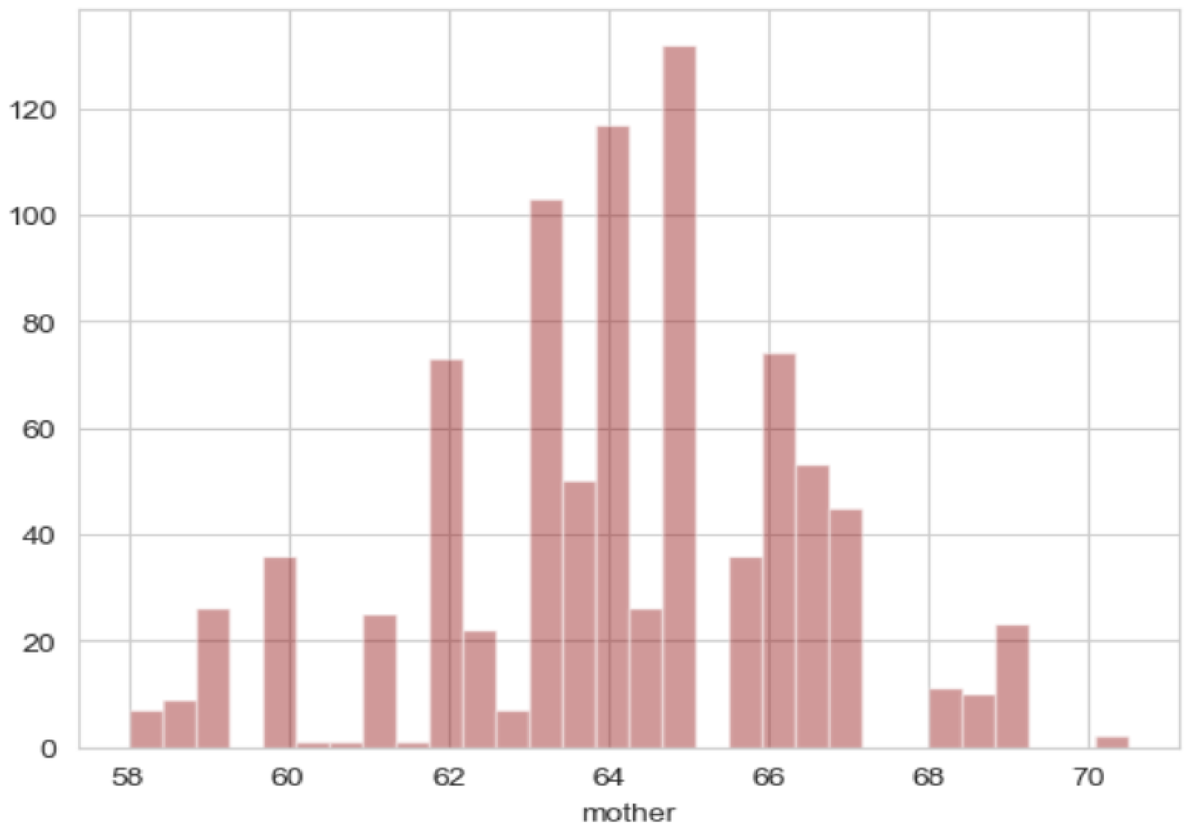


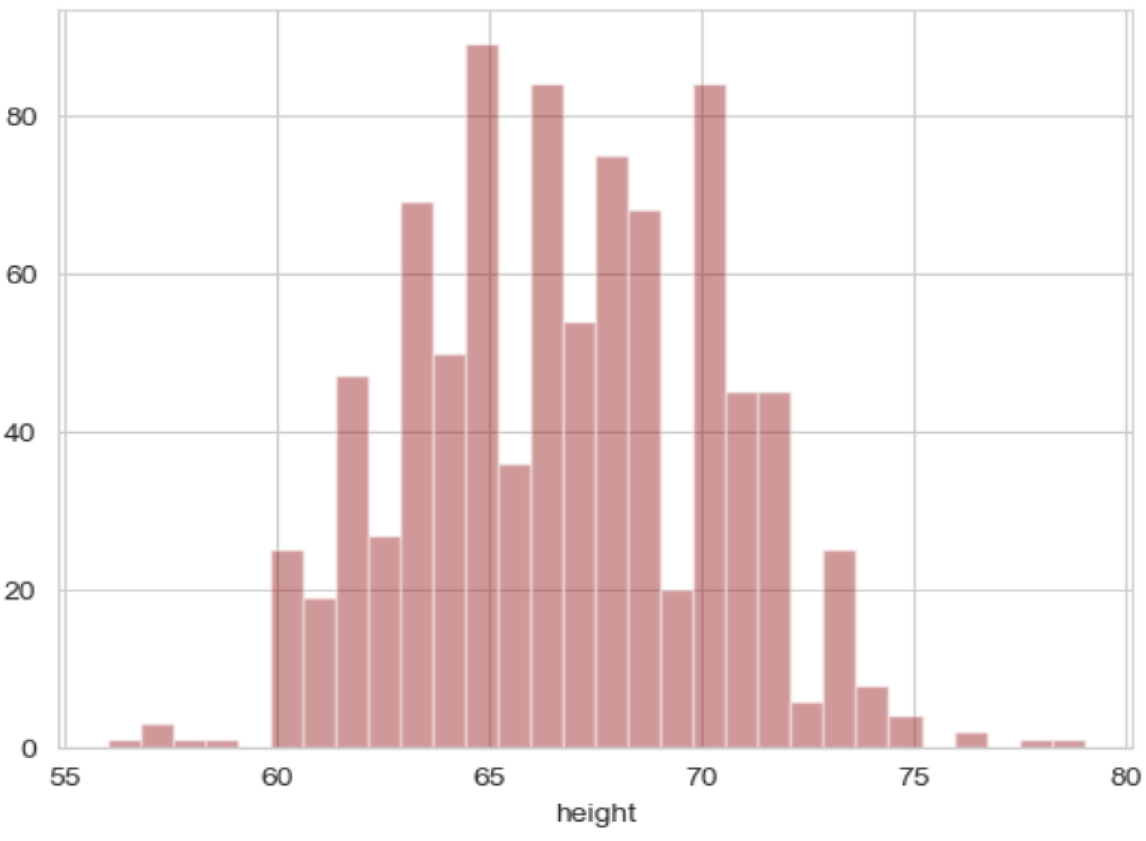
Scatter plot of mother and child height.



Boxplot of child height and Gender.




0
20
40
60
80
100
120
58
60
62
64
66
68
70
mother


0
20
40
60
80
55
60
65
70
75
80
height

Model Building

Splitting data for training and testing purpose

We split the given train dataset into two parts for training and testing purpose. The split ratio we used is 0.8 which indicates we used 80% data for training purpose and 20% data for testing purpose. We will be using the same split ratio for all the models trained.

Now we will be training our required models. Our project goal requires us to train specific 4 Regression models viz.

1. Linear Regression
2. Random Forest Regression
3. Decision Tree Regression
4. Gradient boosting regressor

Linear Regression

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the predictor variables and the response variable. The objective of linear regression is to find the best-fitting straight line (or hyperplane in multiple dimensions) that minimizes t he sum of squared differences between the predicted and actual values.

In simple linear regression, there is a single independent variable that is used to pre dict the dependent variable. The relationship is represented by a linear equation of the form $Y = β0 + β1*X + ε$, where Y is the dependent variable, X is the independent variable, β0 is the intercept, β1 is the coefficient for the independent variable, and ε is the error term.

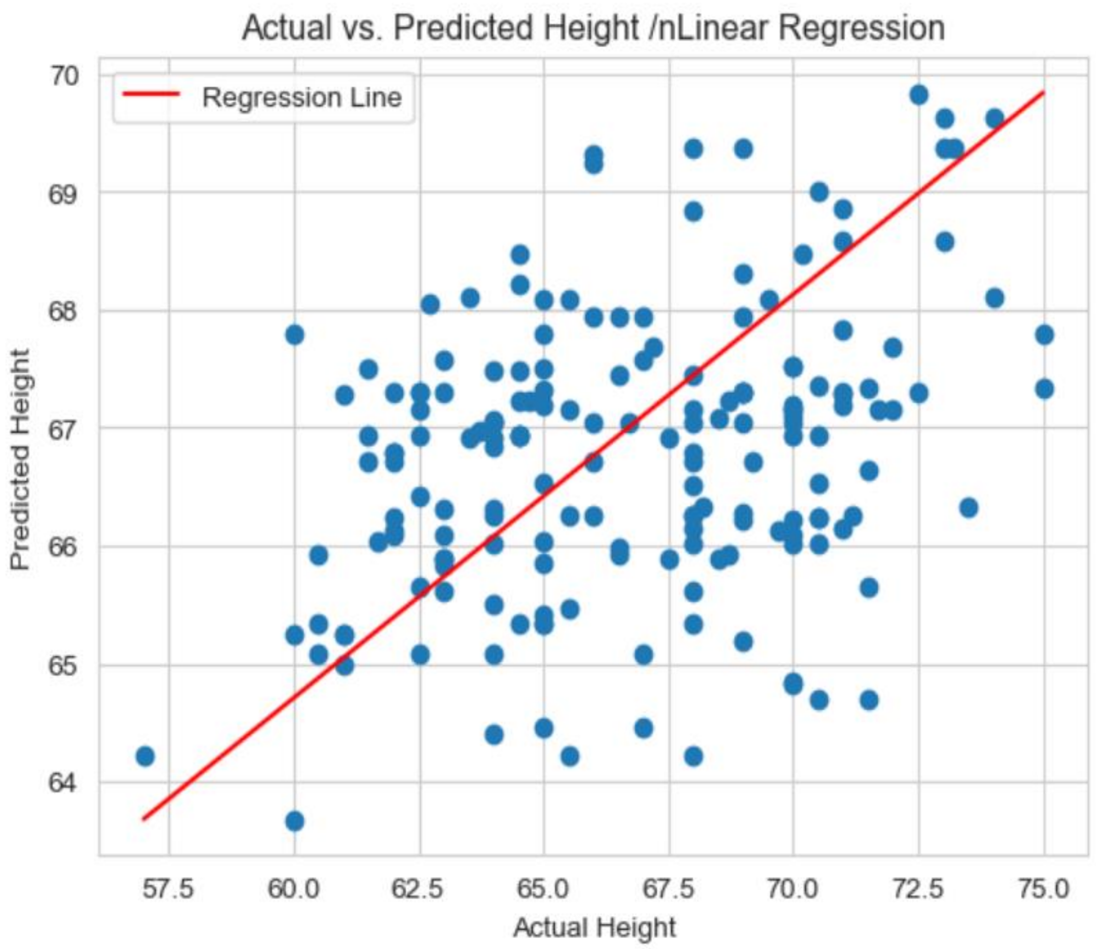
Multiple linear regression extends the concept to include multiple independent vari ables. The equation becomes $Y = β0 + β1*X1 + β2*X2 + ... + βn*Xn + ε$, where X1, X2, ..., Xn a re the independent variables, β1, β2, ..., βn are their corresponding coefficients, and ε is the error term.

The coefficients in linear regression represent the change in the dependent variable for a one-unit change in the corresponding independent variable, assuming all other variable s are held constant. These coefficients provide insights into the strength and direction of the relationship between the independent variables and the dependent variable.

Linear regression models can be evaluated using various metrics, such as mean squ ared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), coefficient o f determination (R-squared), and adjusted R-squared. These metrics help assess the perform ance and accuracy of the model in predicting the dependent variable.

Linear regression has broad applications in fields such as statistics, economics, finan ce, social sciences, and machine learning. It provides a straightforward approach to understa nd and quantify relationships between variables, making it a valuable tool for analysis and pr ediction tasks.

```
Mean Squared Error: 12.01999646629591
Root Mean Squared Error: 3.4669866550501616
R-squared Score: 0.11584237747872972
Regression Line Equation: y = 0.39x + 23.39
```



Random Forest Regression

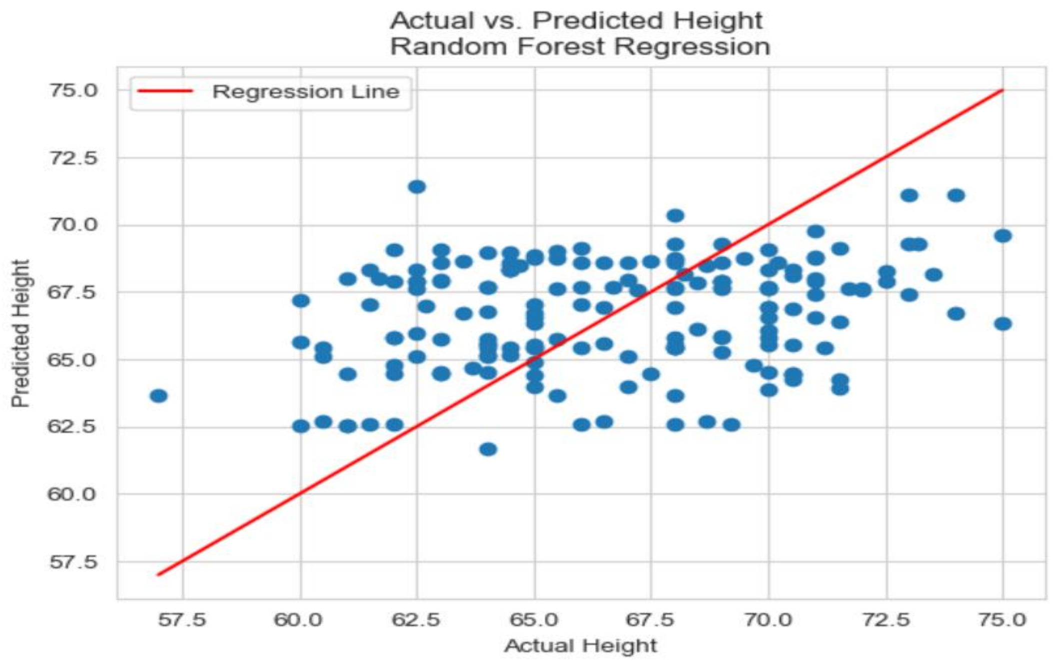
Random Forest Regression is a powerful machine learning algorithm used for regres sion tasks. It is an extension of the Random Forest algorithm, which combines multiple decisi on trees to make more accurate predictions.

In Random Forest Regression, a collection of decision trees is constructed, where ea ch tree is trained on a random subset of the training data and a random subset of the feature s. During training, each decision tree predicts the target variable based on a set of input feat ures. The final prediction is obtained by averaging the predictions of all the individual trees in the forest

Random Forest Regression is suitable for a wide range of regression problems, espe cially when dealing with complex data and high-dimensional feature spaces. It can handle bo th continuous and categorical features and can effectively capture non-linear relationships be tween the input variables and the target variable.During prediction, the Random Forest Regr ession model averages the predictions from all the individual trees, providing a robust and re liable estimate of the target variable.

In summary, Random Forest Regression is an ensemble learning technique that com bines multiple decision trees to make accurate predictions in regression tasks. It overcomes t he limitations of individual decision trees and provides a powerful and flexible approach for r egression modeling.

```
Mean Squared Error: 13.736538061508025
Root Mean Squared Error: 3.706283591619511
R-squared Score: -0.010421664281784038
```



Decision Tree Regression

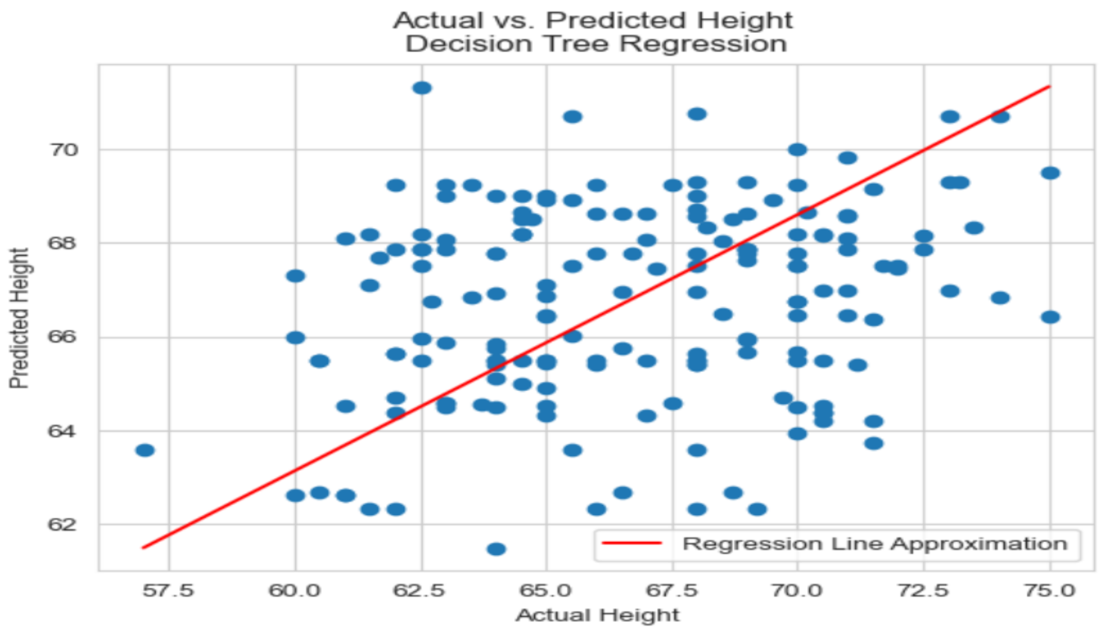
Decision Tree Regression is a machine learning algorithm used for regression tasks. It is a type of decision tree-based algorithm that models the relationship between input features and the target variable by recursively partitioning the data into smaller subsets.

In Decision Tree Regression, the algorithm builds a tree-like model where each internal node represents a feature and a splitting criterion, and each leaf node represents a predicted value. The goal is to minimize the overall prediction error by creating decision rules that divide the data into homogeneous regions.

The algorithm constructs the decision tree in a top-down manner by selecting the best feature and splitting criterion at each step. The quality of the split is typically evaluated using metrics such as mean squared error (MSE) or mean absolute error (MAE). The splitting process continues until a stopping criterion is met, such as reaching a maximum depth, having a minimum number of samples in a leaf node, or when further splitting does not significantly reduce the prediction error.

During the prediction phase, an unseen data point traverses the decision tree based on the feature values. It follows the decision rules from the root to a specific leaf node, and the predicted value at that leaf node is assigned as the output.

```
Mean Squared Error: 14.026768855957972
Root Mean Squared Error: 3.7452328173236404
R-squared Score: -0.03177023704741311
```



Gradient Boosting Regressor

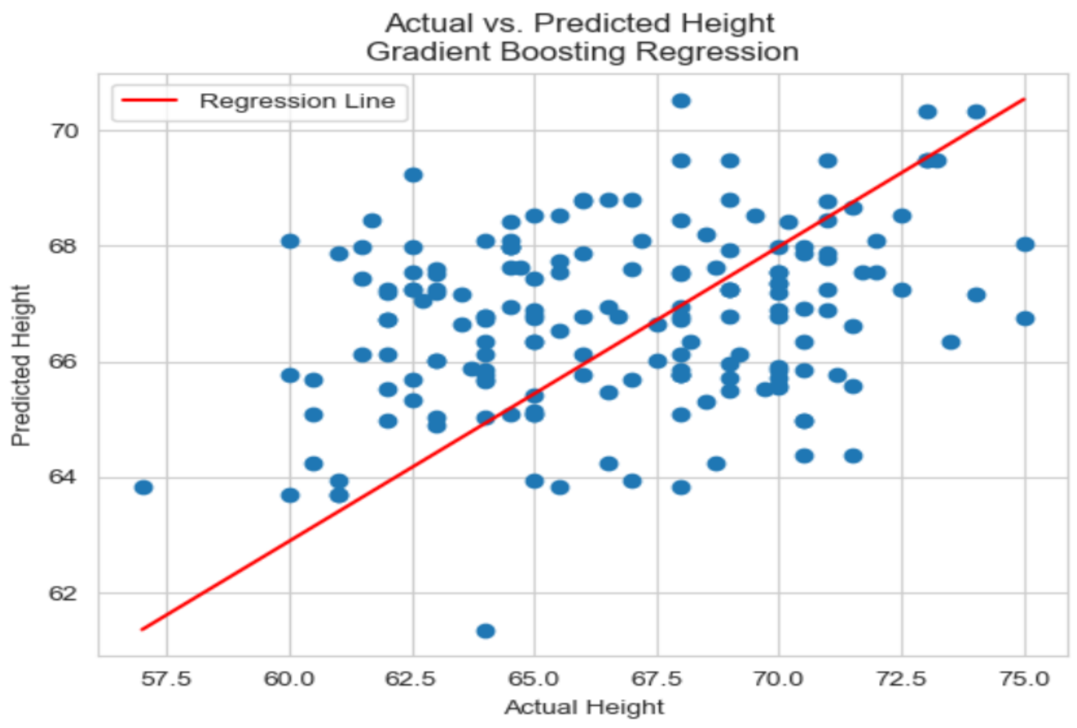
Gradient Boosting Regressor is a machine learning algorithm used for regression tas ks. It is a boosting algorithm that combines multiple weak learners (usually decision trees) to create a strong predictive model.

In Gradient Boosting Regressor, the algorithm is built in an iterative manner. Initially , a weak learner, often a shallow decision tree, is fitted to the data. The subsequent iterations focus on improving the predictions by adding new weak learners that are trained to correct t he errors made by the previous models. Each new weak learner is fitted on the residuals (the differences between the predicted and actual values) of the previous models.

The main idea behind gradient boosting is to fit new models that minimize the loss function (usually mean squared error) by finding the negative gradient of the loss function wi th respect to the predictions. This allows the algorithm to make smaller and more accurate c orrections with each iteration.

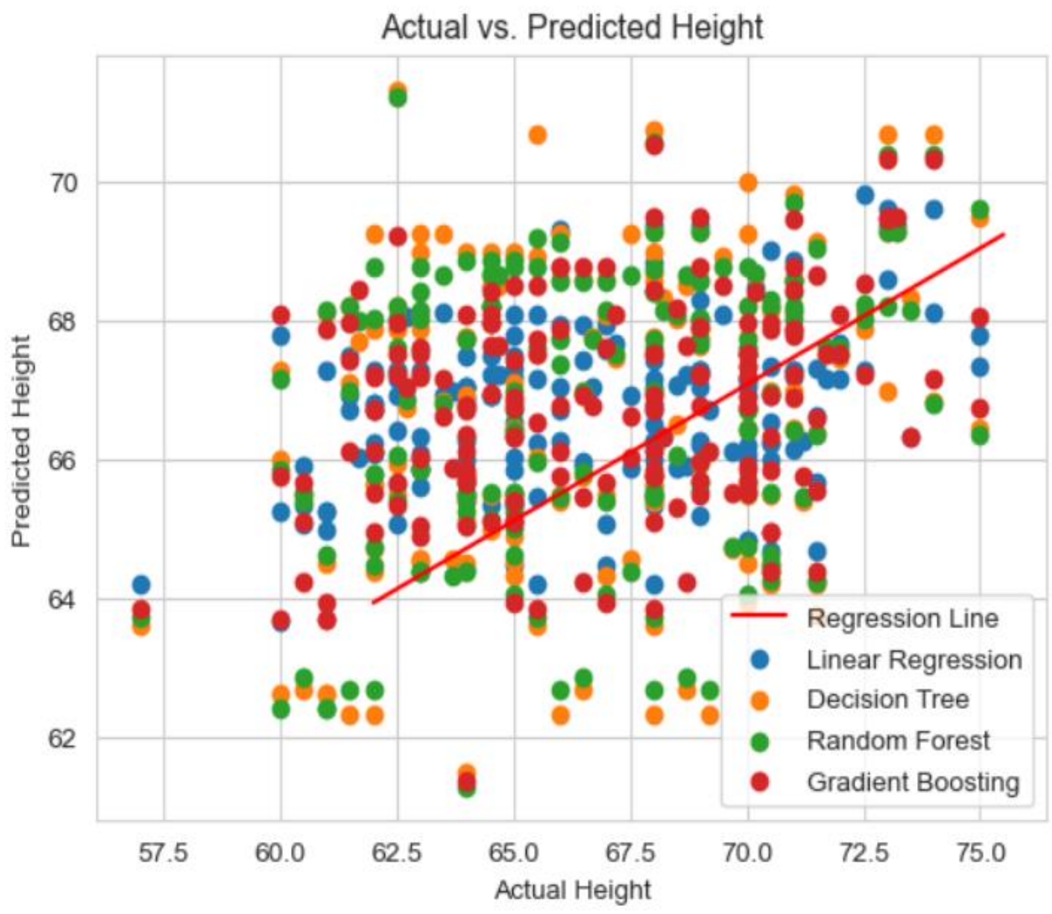
During prediction, the final model combines the predictions of all the weak learners to provide the overall prediction. The number of iterations and the learning rate are hyperpa rameters that control the complexity and generalization of the model.

```
Mean Squared Error: 12.48694800134241
Root Mean Squared Error: 3.5336875924934863
R-squared Score: 0.08149471687691268
The accuracy of our model is 8.15%
```

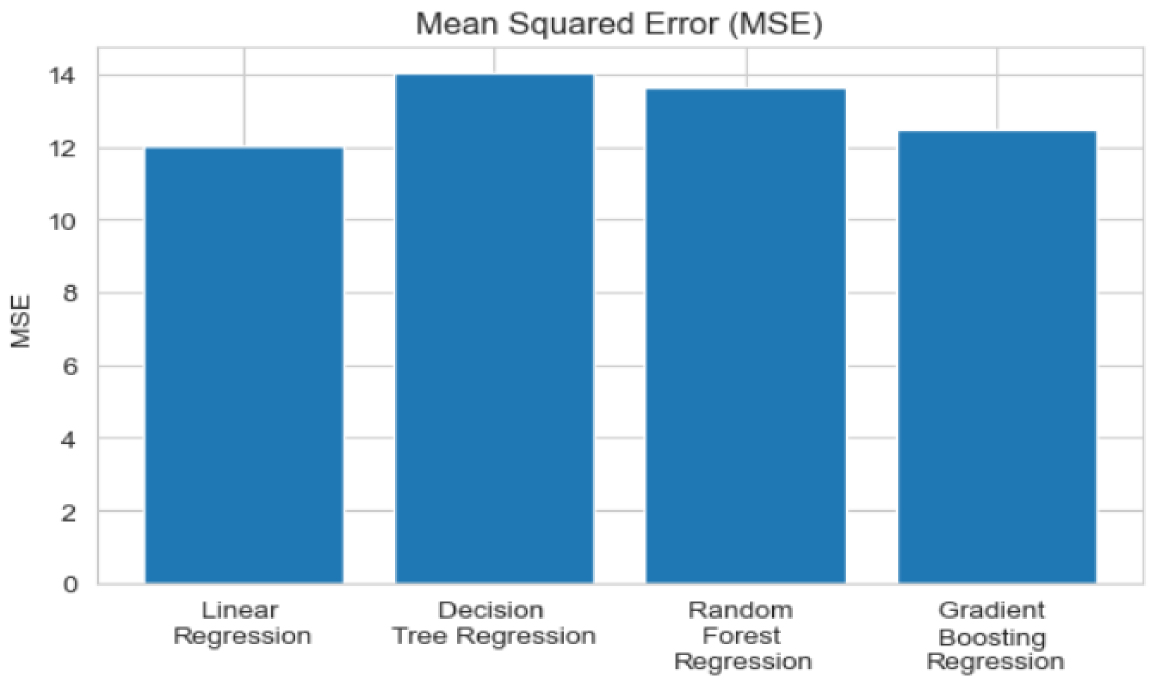


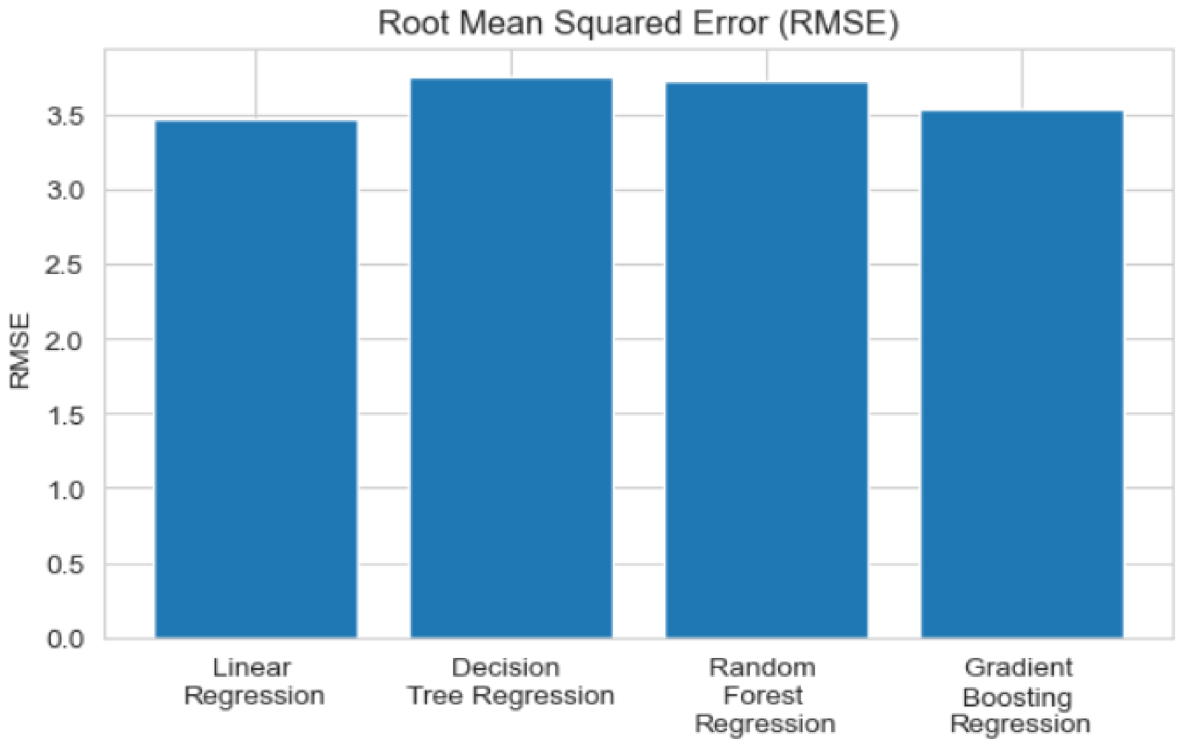
Comparison of the Models trained

```
Linear Regression: y = 0.39 * x1 + 0.25 * x2 + 23.39
Decision Tree: Equation not available for non-linear models.
Random Forest: Equation not available for non-linear models.
Gradient Boosting: Equation not available for non-linear models.
```

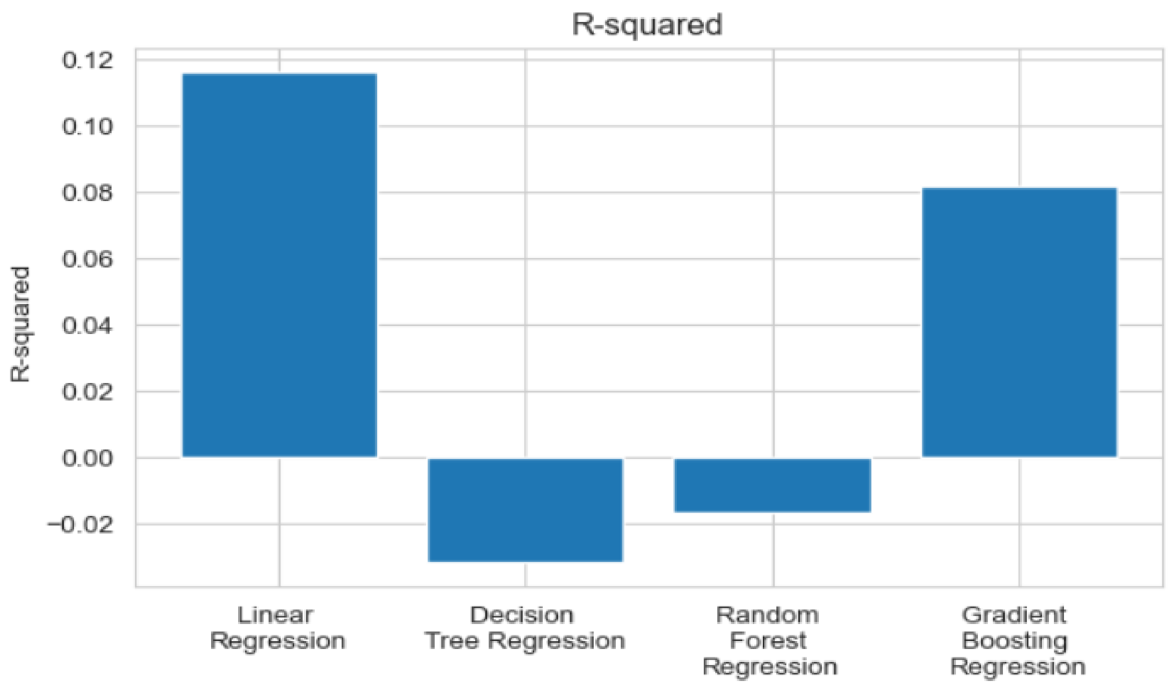


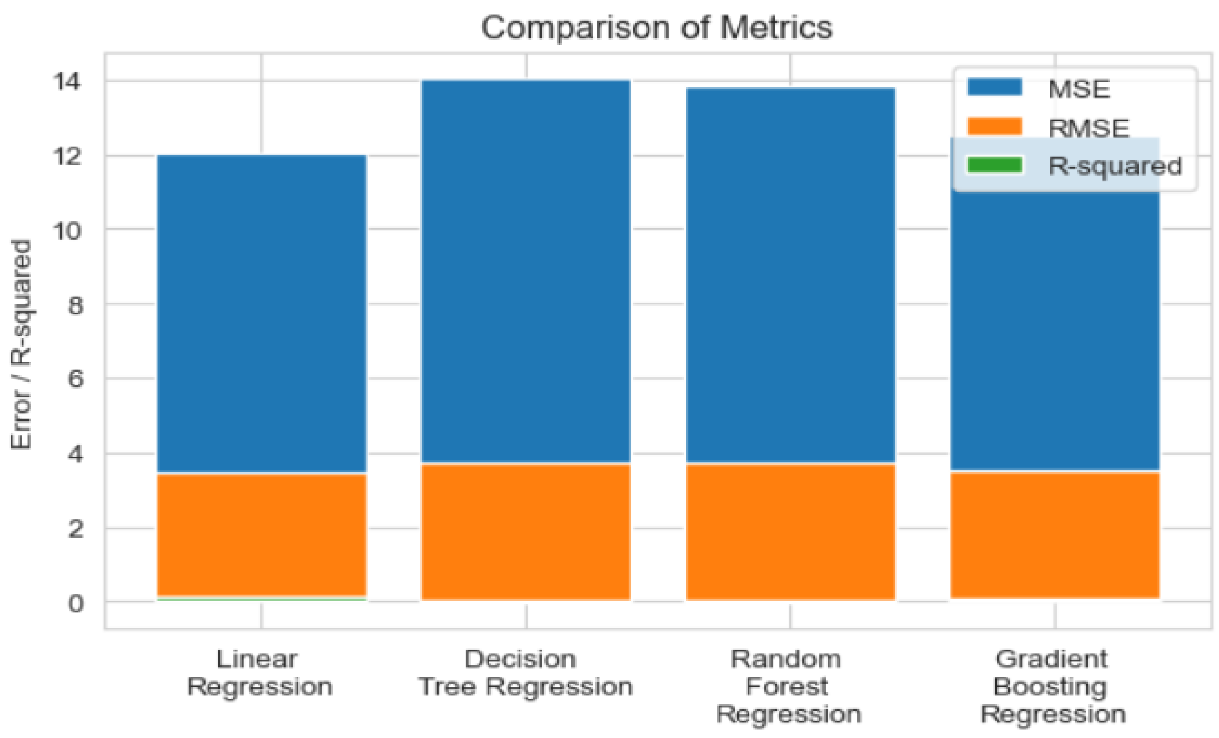
```
Linear Regression: R-squared Score = 0.11584237747872972
Decision Tree: R-squared Score = -0.03177023704741333
Random Forest: R-squared Score = -0.001834604879242585
Gradient Boosting: R-squared Score = 0.08149471687691268
```


Mean Squared Error (MSE)
MSE
0
2
4
6
8
10
12
14
Linear Regression
Decision Tree Regression
Random Forest Regression
Gradient Boosting Regression

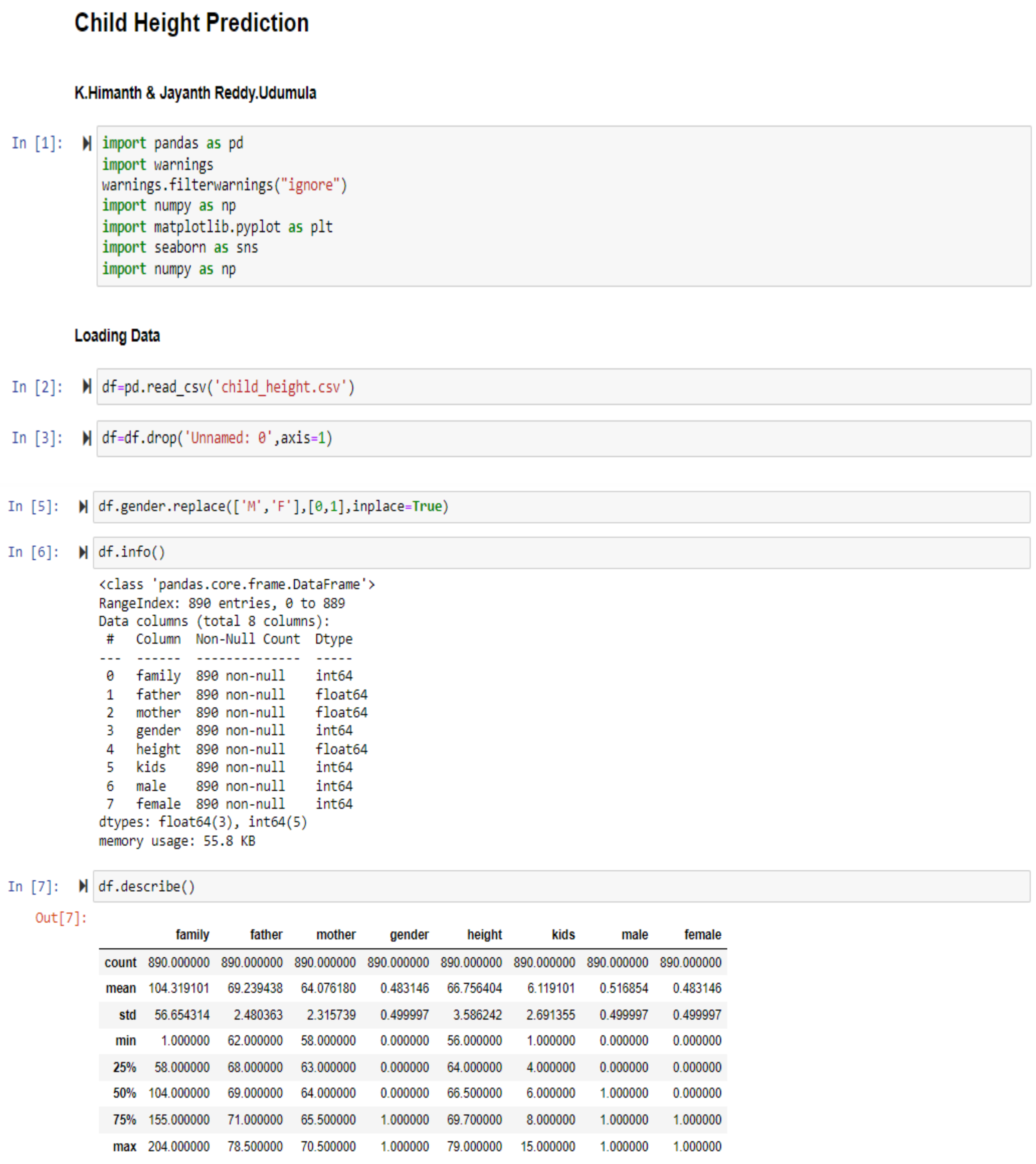

Root Mean Squared Error (RMSE)
RMSE
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Linear Regression
Decision Tree Regression
Random Forest Regression
Gradient Boosting Regression

Regression
Regression


R-squared
R-squared
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
Linear Regression
Decision Tree Regression
Random Forest Regression
Gradient Boosting Regression


Comparison of Metrics
Error / R-squared
14
12
10
8
6
4
2
0
MSE
RMSE
R-squared
Linear Regression
Decision Tree Regression
Random Forest Regression
Gradient Boosting Regression

Codes



Child Height Prediction

K.Himanth & Jayanth Reddy.Udumula

In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Loading Data

In [2]:
```
df=pd.read_csv('child_height.csv')
```

In [3]:
```
df=df.drop('Unnamed: 0',axis=1)
```

In [5]:
```
df.gender.replace(['M','F'],[0,1],inplace=True)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 890 entries, 0 to 889
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   family  890 non-null    int64
 1   father  890 non-null    float64
 2   mother  890 non-null    float64
 3   gender  890 non-null    int64
 4   height  890 non-null    float64
 5   kids    890 non-null    int64
 6   male    890 non-null    int64
 7   female  890 non-null    int64
dtypes: float64(3), int64(5)
memory usage: 55.8 KB
```

In [7]:
```
df.describe()
```

Out[7]:

	family	father	mother	gender	height	kids	male	female
count	890.000000	890.000000	890.000000	890.000000	890.000000	890.000000	890.000000	890.000000
mean	104.319101	69.239438	64.076180	0.483146	66.756404	6.119101	0.516854	0.483146
std	56.654314	2.480363	2.315739	0.499997	3.586242	2.691355	0.499997	0.499997
min	1.000000	62.000000	58.000000	0.000000	56.000000	1.000000	0.000000	0.000000
25%	58.000000	68.000000	63.000000	0.000000	64.000000	4.000000	0.000000	0.000000
50%	104.000000	69.000000	64.000000	0.000000	66.500000	6.000000	1.000000	0.000000
75%	155.000000	71.000000	65.500000	1.000000	69.700000	8.000000	1.000000	1.000000
max	204.000000	78.500000	70.500000	1.000000	79.000000	15.000000	1.000000	1.000000

```
In [8]: df.isnull()
```

Out[8]:

	family	father	mother	gender	height	kids	male	female
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
885	False	False	False	False	False	False	False	False
886	False	False	False	False	False	False	False	False
887	False	False	False	False	False	False	False	False
888	False	False	False	False	False	False	False	False
889	False	False	False	False	False	False	False	False

890 rows × 8 columns

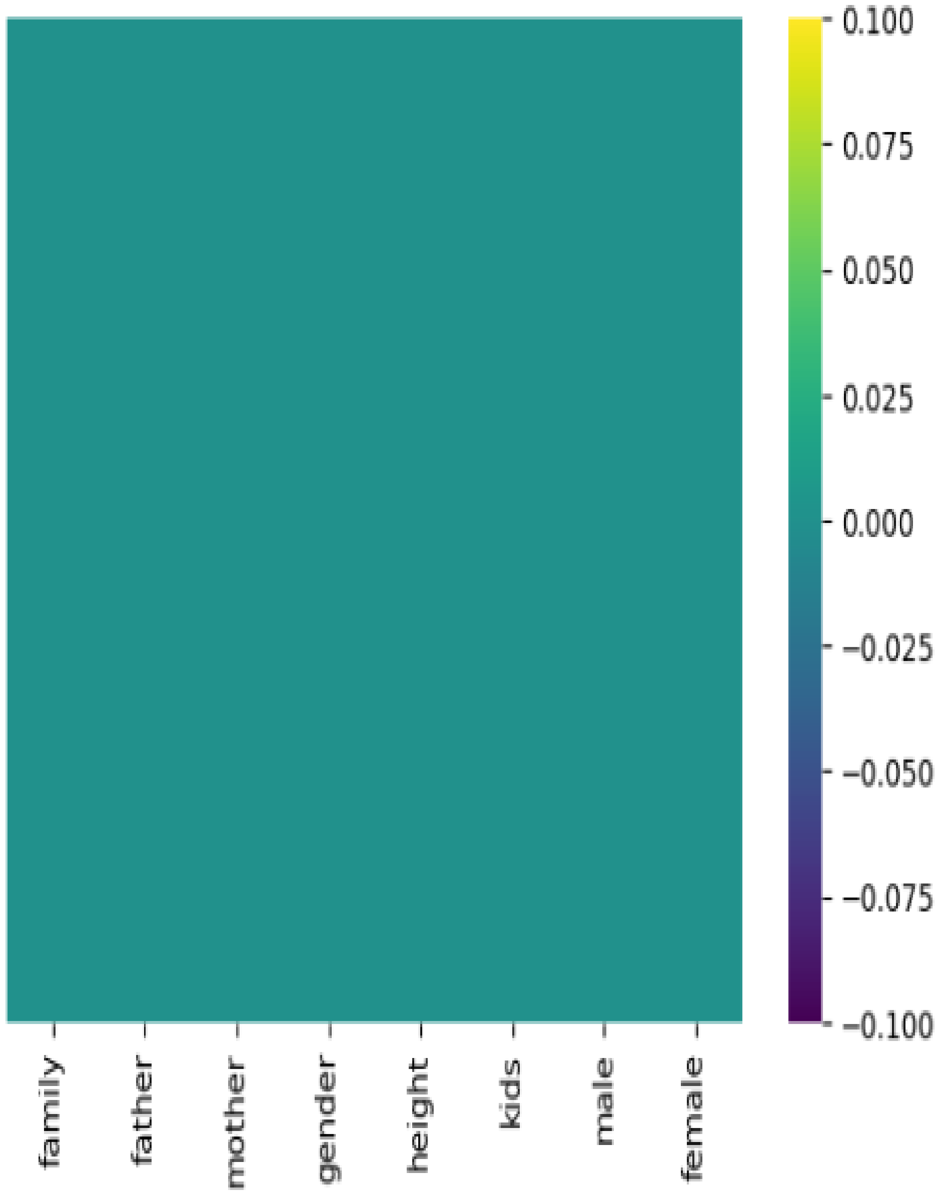
```
In [9]: print("Null values couunt:\n",df.isnull().sum())
```

```
Null values couunt:
 family    0
father    0
mother    0
gender    0
height    0
kids      0
male      0
female    0
dtype: int64
```

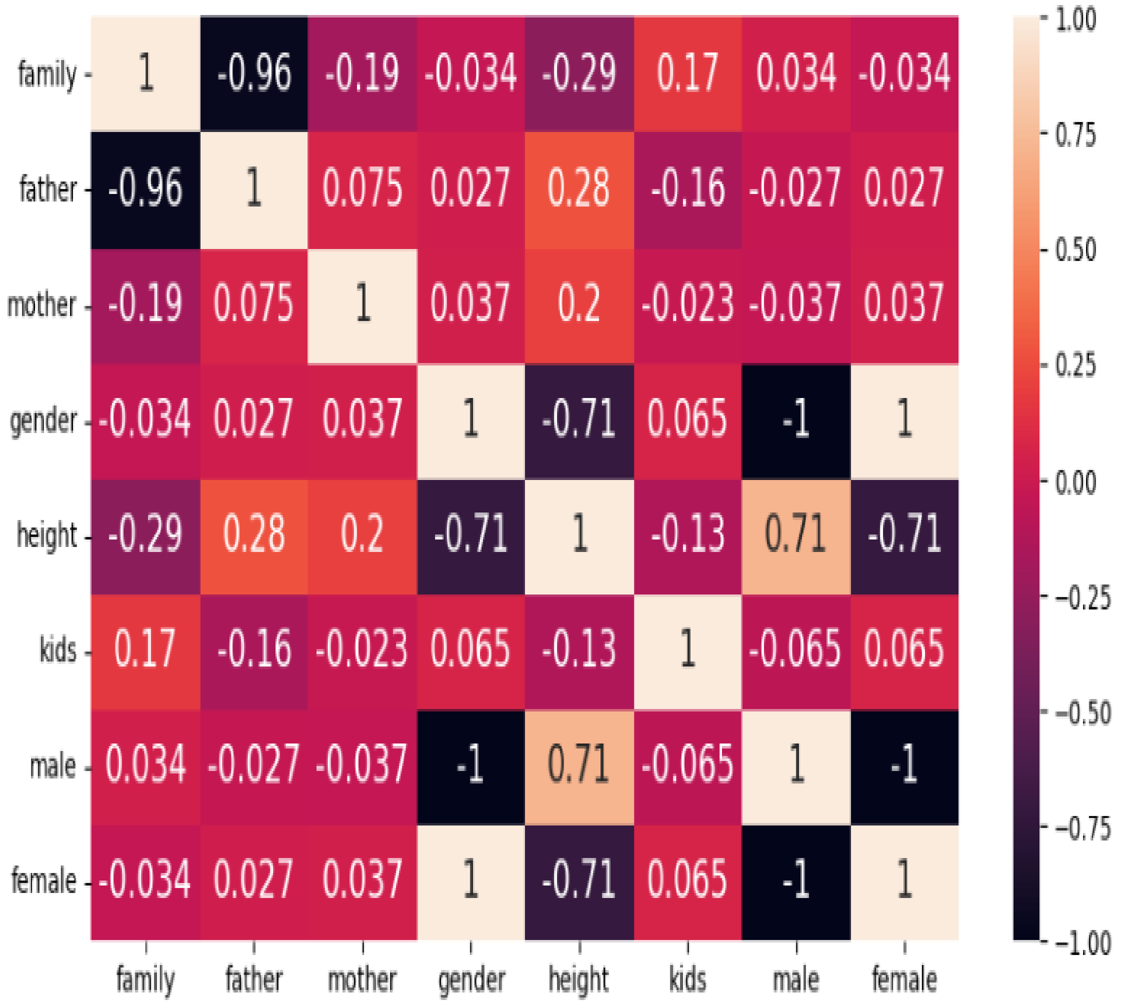
```
In [10]: f=df.columns[1:-1]
         print("\n correltion of each attribute with height:\n")
         for i in f:
             corr=df[i].corr(df['height'])
             print(i,":",corr)
```

```

 correltion of each attribute with height:

father : 0.2768008110348267
mother : 0.20203872284119695
gender : -0.7126729743888253
height : 1.0
kids : -0.12847496133255157
male : 0.7126729743888252
```

```
In [11]:  sns.heatmap(df.isnull(), yticklabels = False, cbar = True, cmap = 'viridis')

Out[11]: <AxesSubplot:>
```



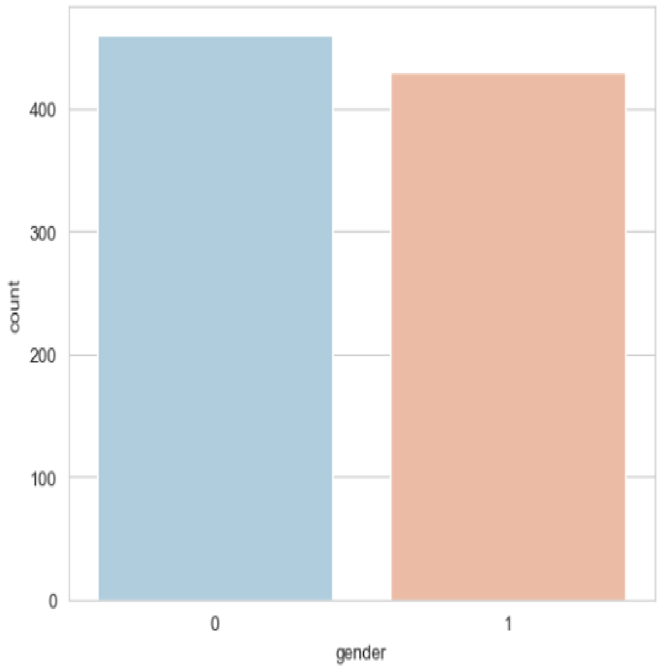
```
In [12]:  plt.figure(figsize = (10, 5))
          sns.heatmap(df.corr(), annot = True, annot_kws = {"size":15})

Out[12]: <AxesSubplot:>
```



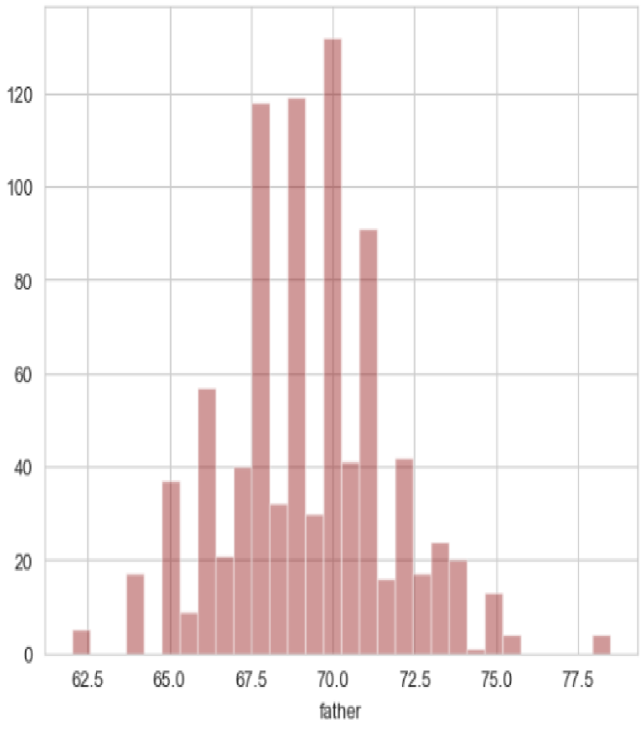
	family	father	mother	gender	height	kids	male	female
family	1	-0.96	-0.19	-0.034	-0.29	0.17	0.034	-0.034
father	-0.96	1	0.075	0.027	0.28	-0.16	-0.027	0.027
mother	-0.19	0.075	1	0.037	0.2	-0.023	-0.037	0.037
gender	-0.034	0.027	0.037	1	-0.71	0.065	-1	1
height	-0.29	0.28	0.2	-0.71	1	-0.13	0.71	-0.71
kids	0.17	-0.16	-0.023	0.065	-0.13	1	-0.065	0.065
male	0.034	-0.027	-0.037	-1	0.71	-0.065	1	-1
female	-0.034	0.027	0.037	1	-0.71	0.065	-1	1

```
In [13]:  sns.set_style('whitegrid')
          sns.countplot(x = 'gender', data = df, palette = 'RdBu_r')
```

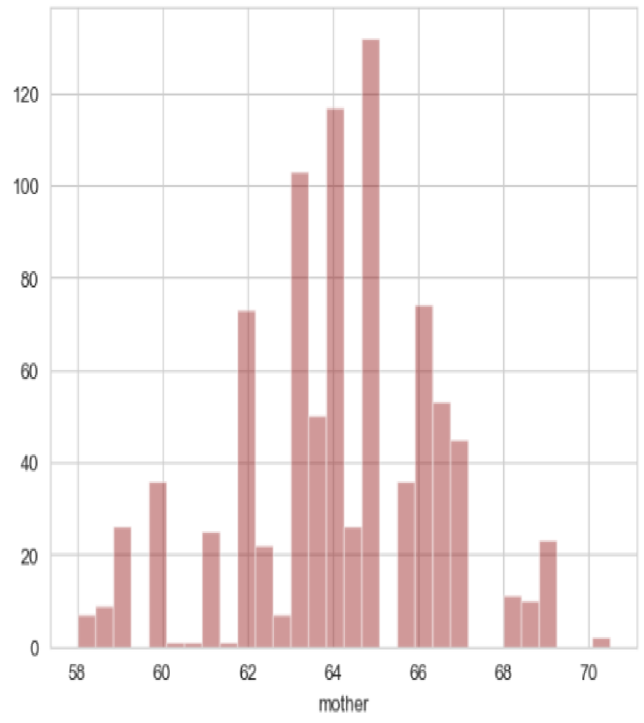
```
Out[13]: <AxesSubplot:xlabel='gender', ylabel='count'>
```



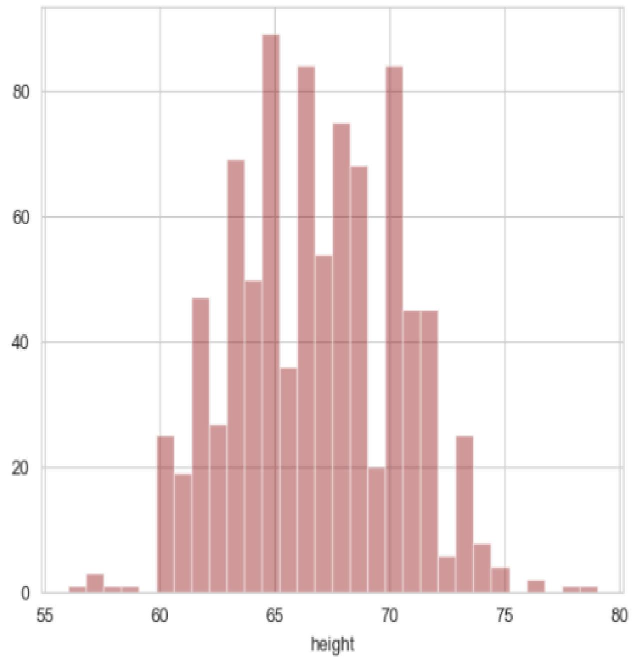
```
In [14]:  sns.distplot(df['father'].dropna(), kde = False, color = 'darkred', bins = 30);
```



In [15]:
```
sns.distplot(df['mother'].dropna(), kde = False, color = 'darkred', bins = 30);
```

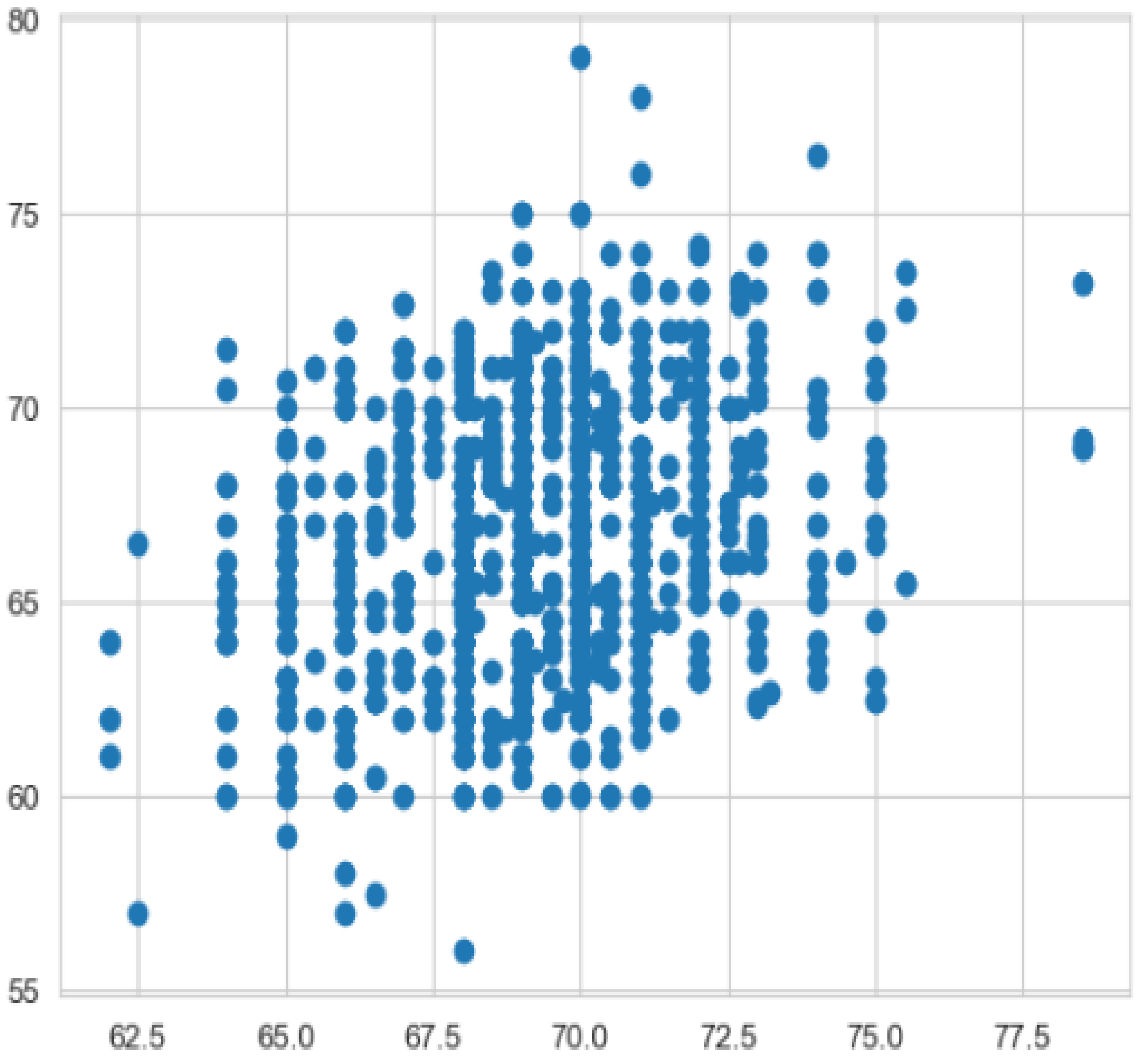


In [16]:
```
sns.distplot(df['height'].dropna(), kde = False, color = 'darkred', bins = 30);
```



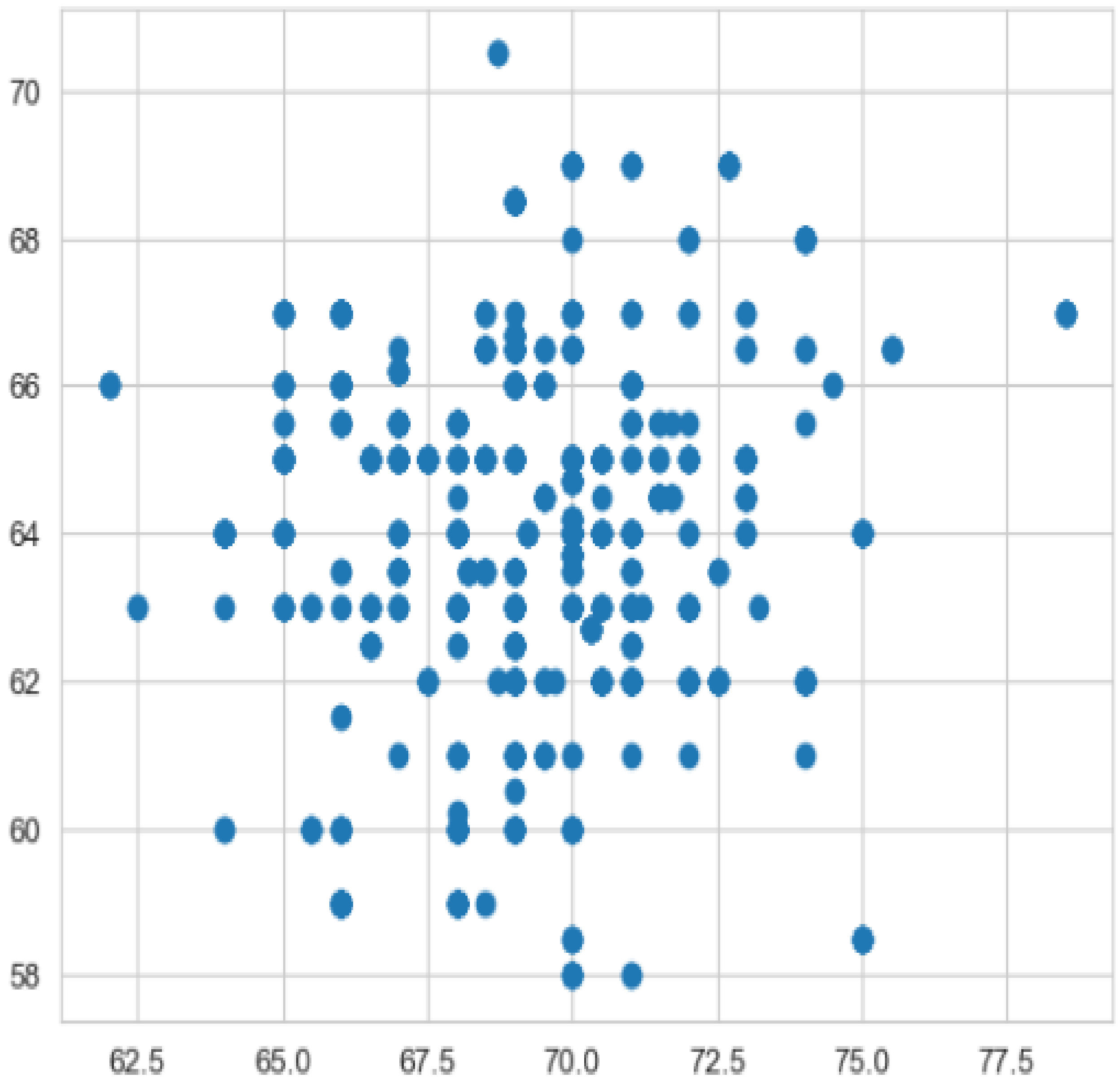
```
plt.scatter(df['father'], df['height'],cmap='viridis')
```

```
3]: <matplotlib.collections.PathCollection at 0x15014324df0>
```



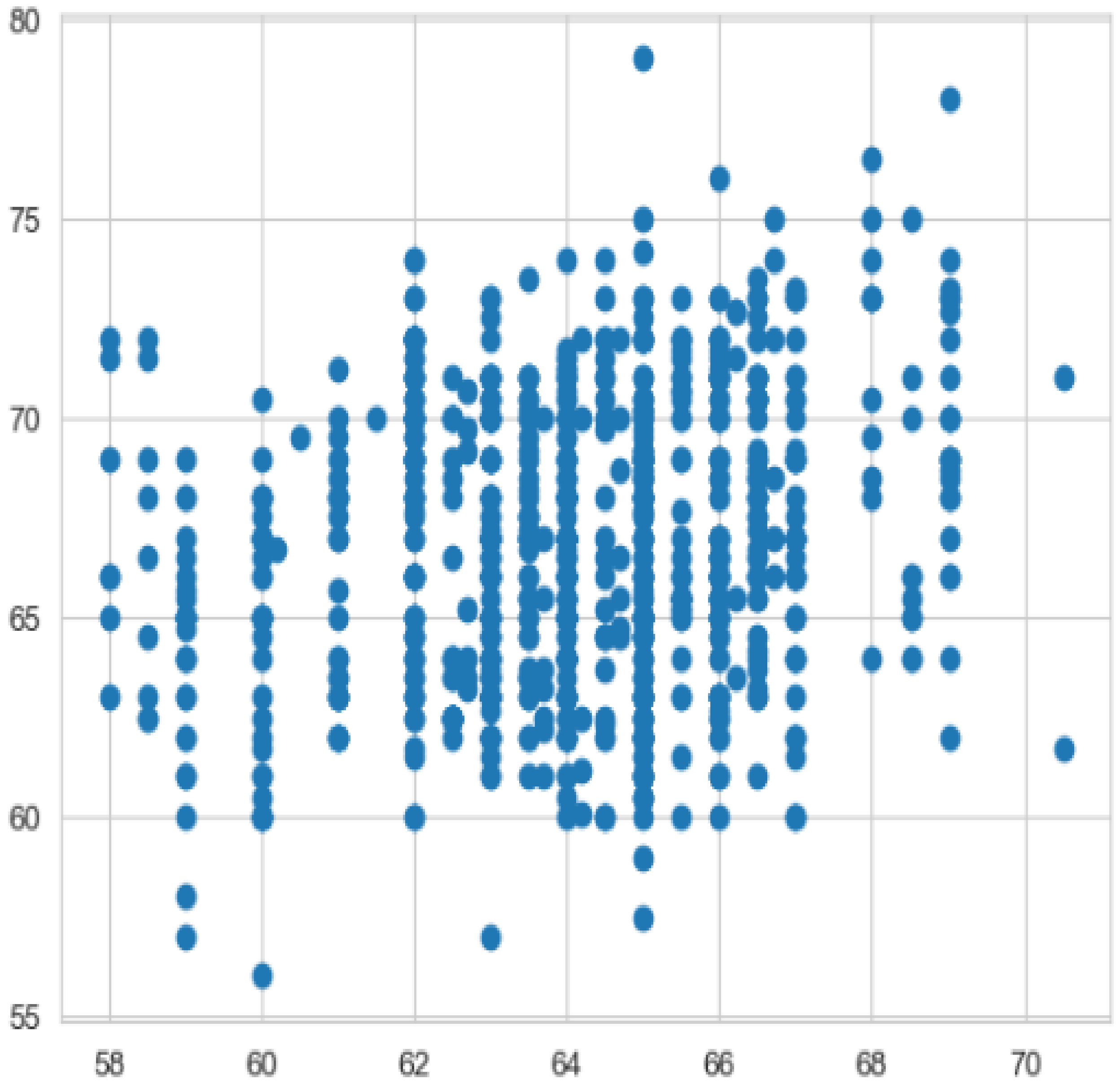
```
plt.scatter(df['father'], df['mother'],cmap='viridis')
```

```
7]: <matplotlib.collections.PathCollection at 0x150140c0640>
```

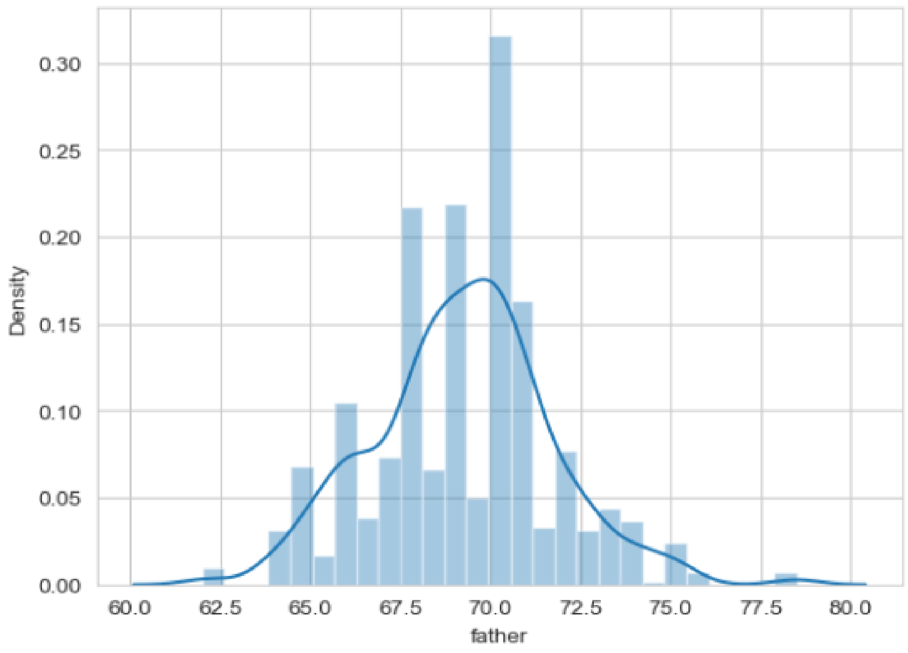


```
plt.scatter(df['mother'], df['height'],cmap='viridis')
```

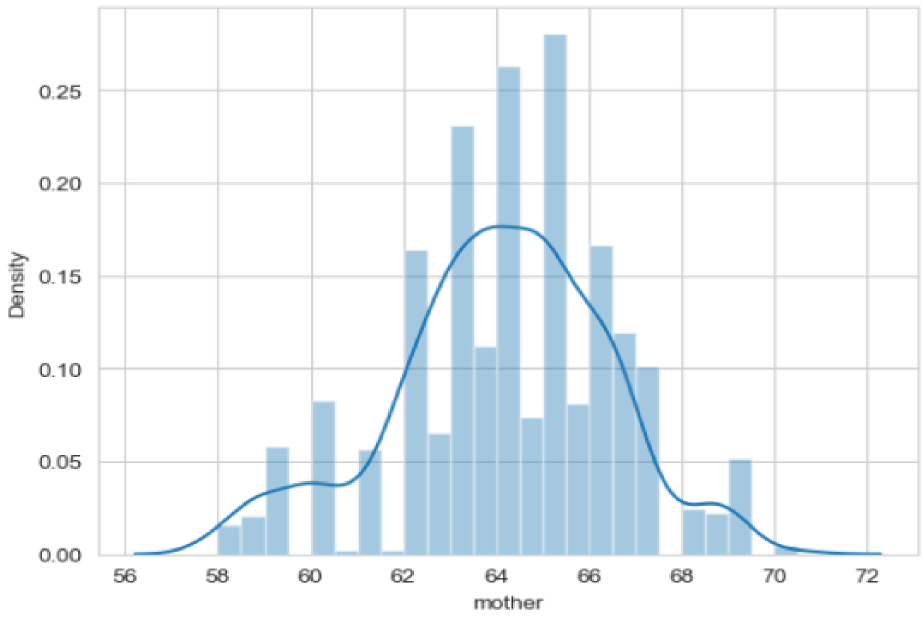
```
]: <matplotlib.collections.PathCollection at 0x1501438d850>
```



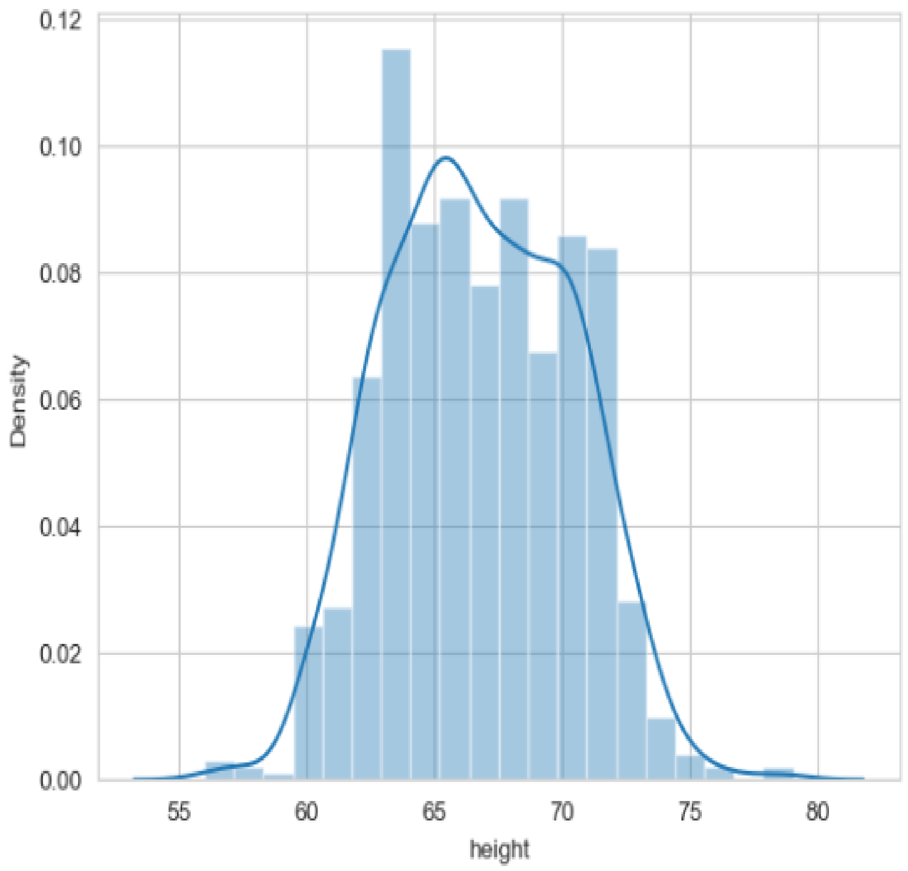
```
In [20]: sns.distplot(df['father'])
Out[20]: <AxesSubplot:xlabel='father', ylabel='Density'>
```



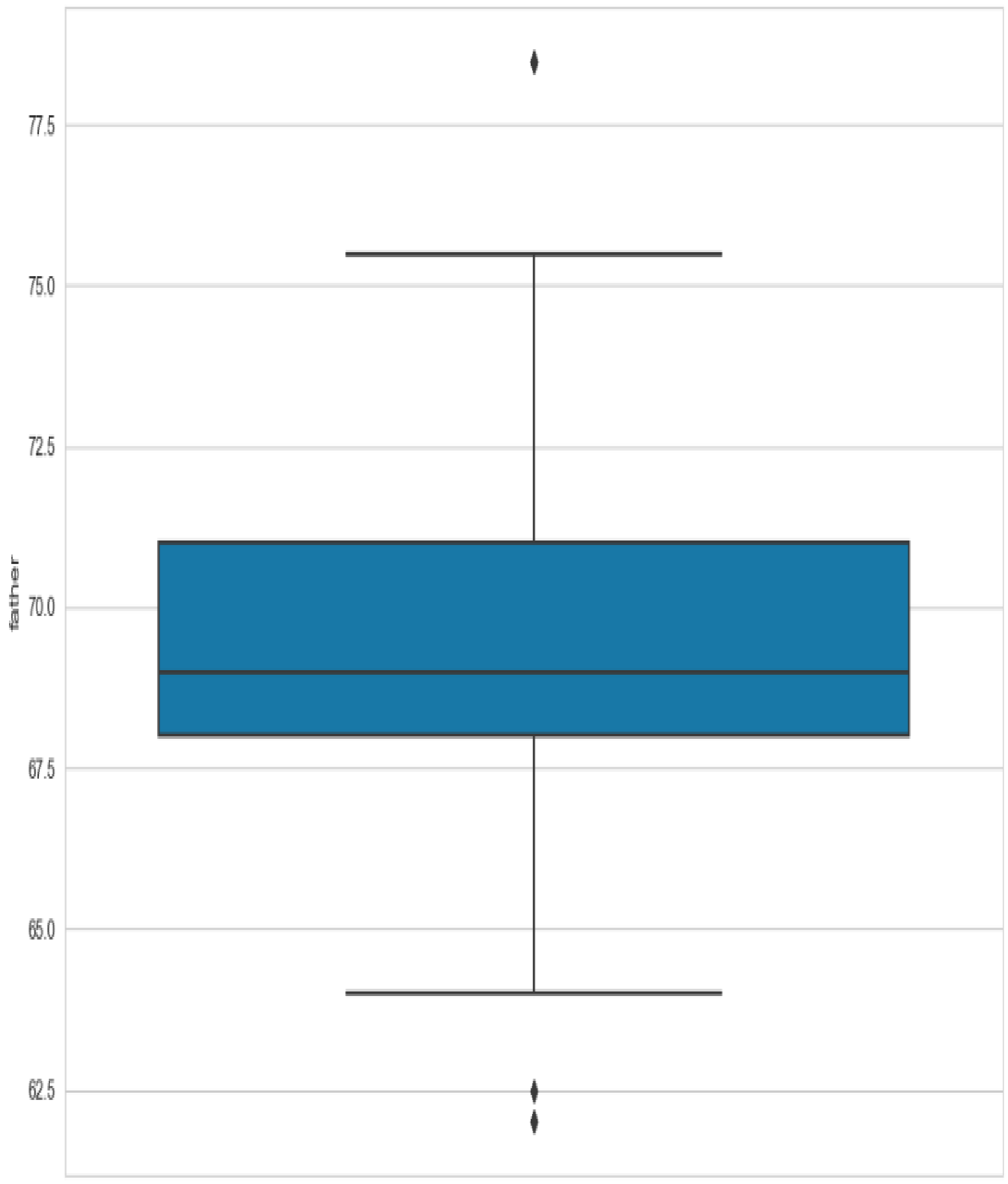
```
In [21]: sns.distplot(df['mother'])
Out[21]: <AxesSubplot:xlabel='mother', ylabel='Density'>
```



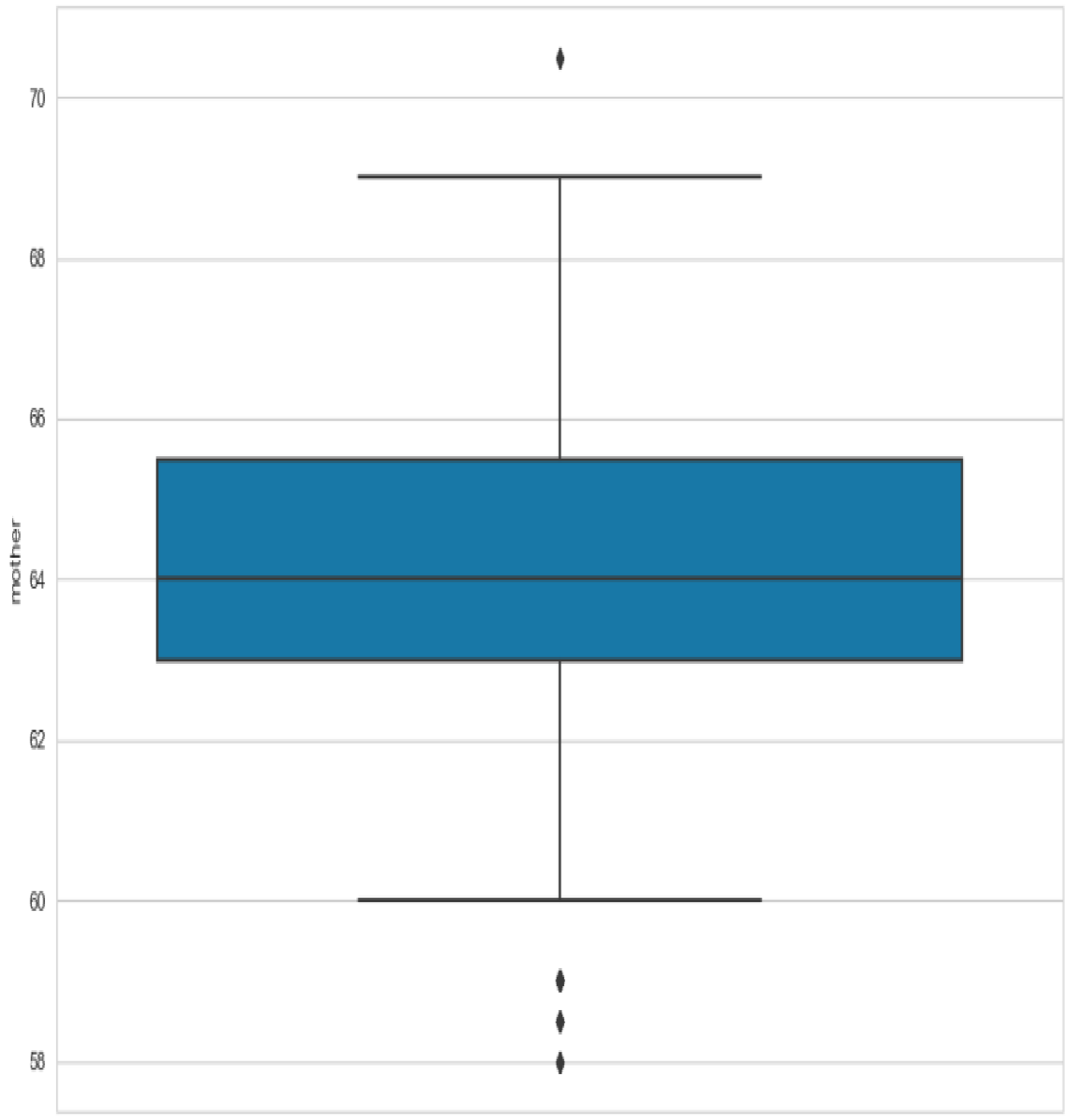
```
In [22]:  sns.distplot(df['height'])

Out[22]: <AxesSubplot:xlabel='height', ylabel='Density'>
```



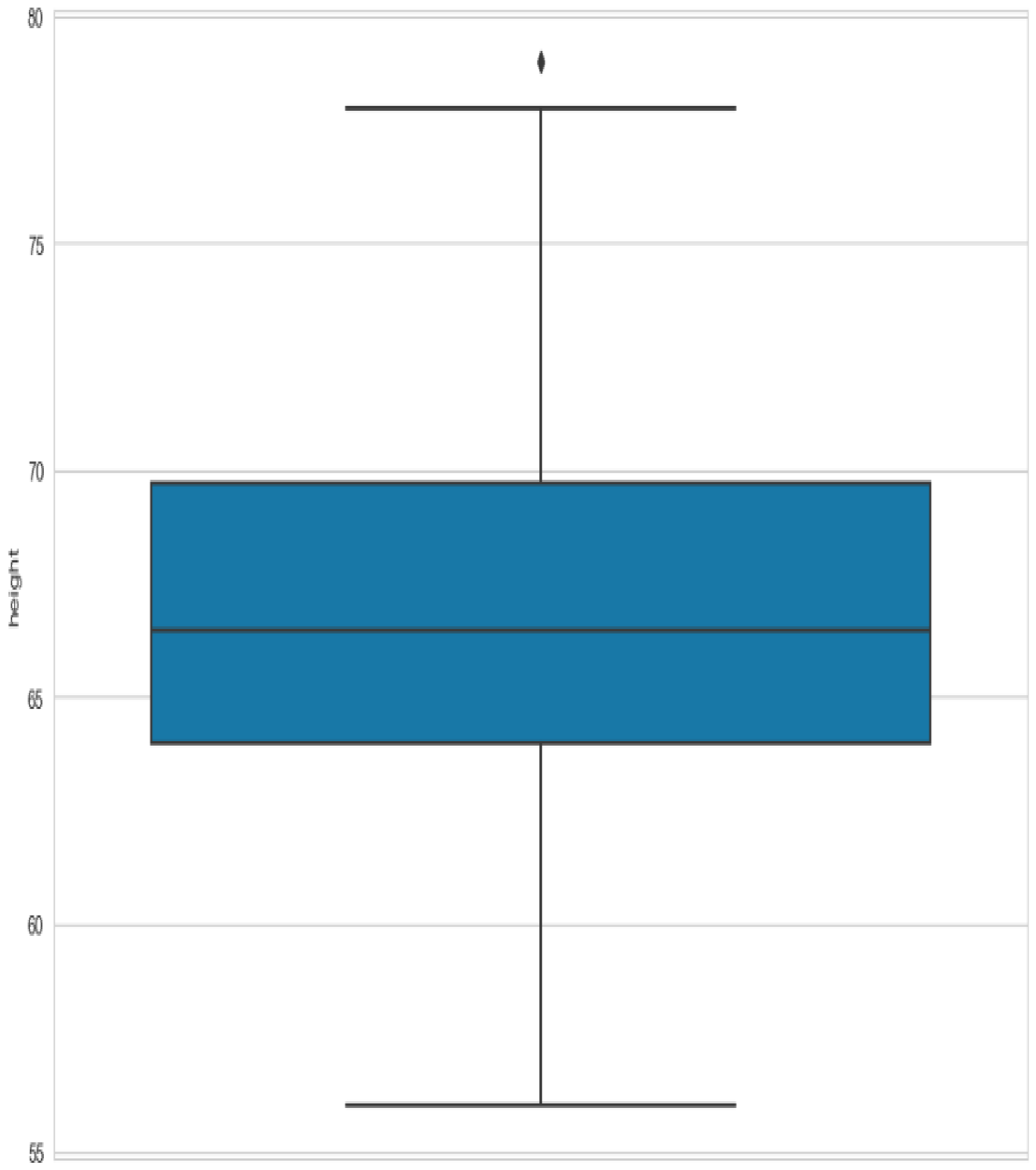
```
In [23]:  plt.figure(figsize=(12,7))
          sns.boxplot(y ='father',  data = df, palette = 'winter')

Out[23]: <AxesSubplot:ylabel='father'>
```



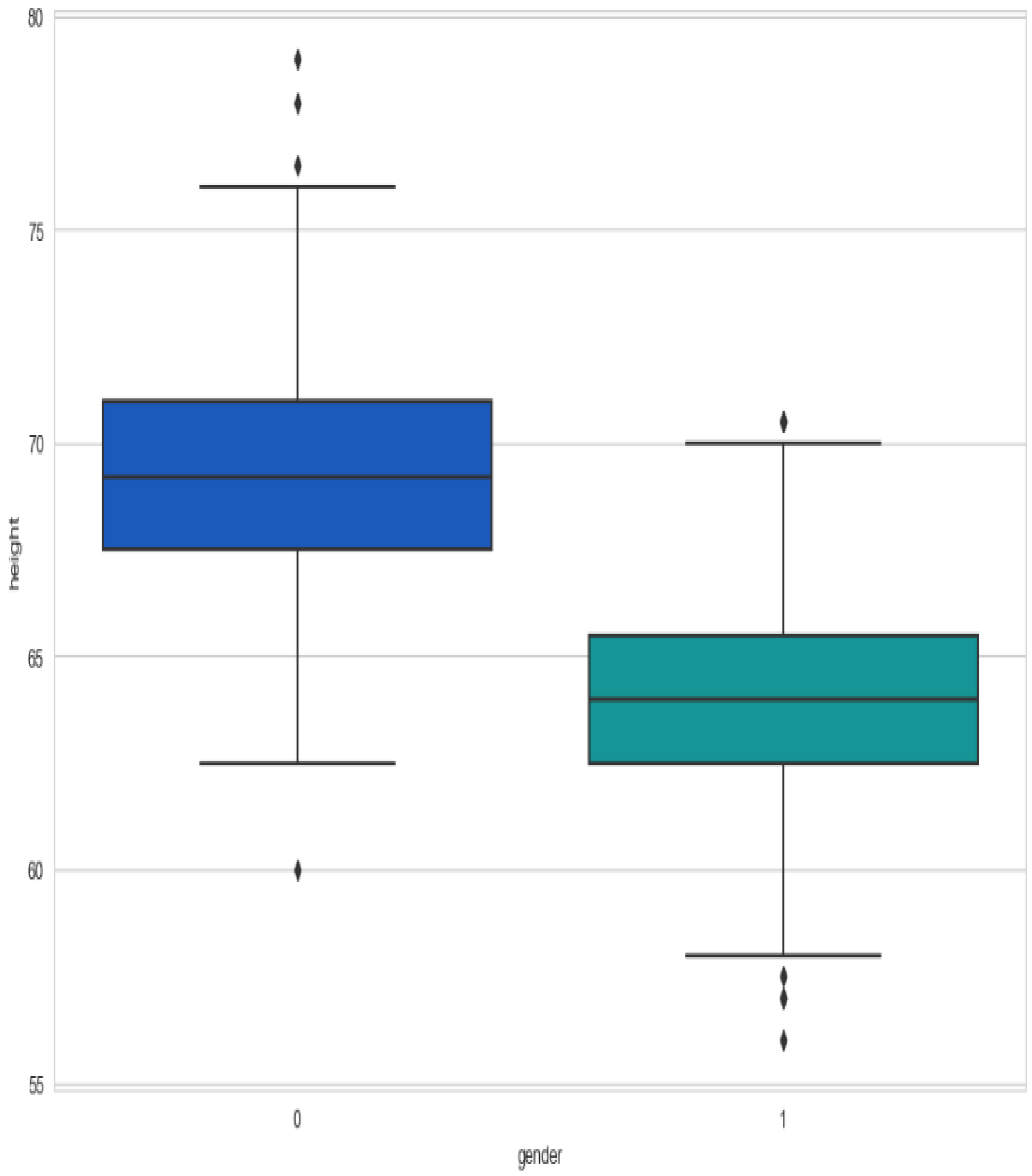
```
In [24]:  plt.figure(figsize = (12, 7))
          sns.boxplot(y = 'mother', data = df, palette = 'winter')

Out[24]: <AxesSubplot:ylabel='mother'>
```

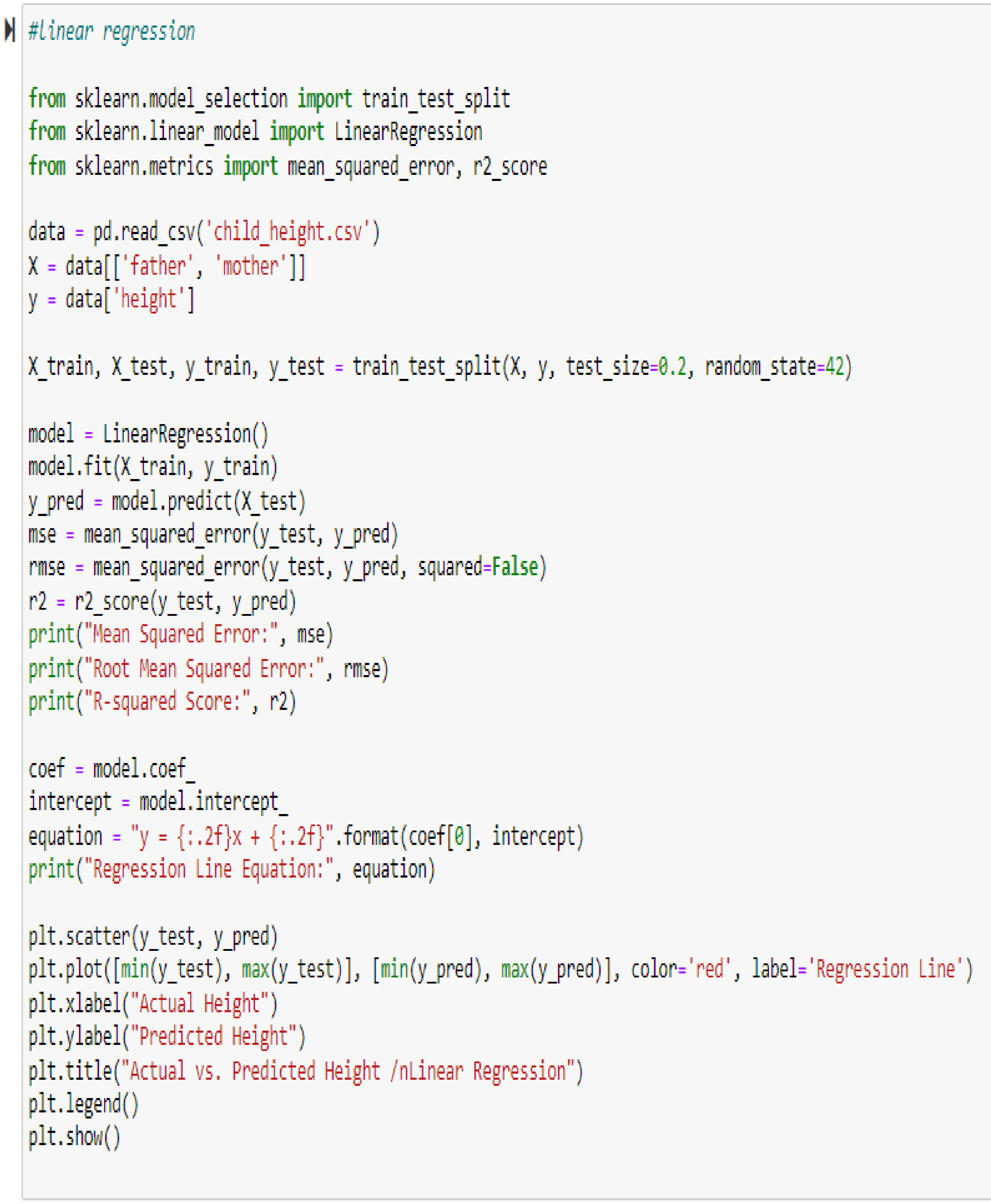


```
In [25]:  plt.figure(figsize = (12, 7))
           sns.boxplot(y = 'height', data = df, palette = 'winter')

Out[25]: <AxesSubplot:ylabel='height'>
```



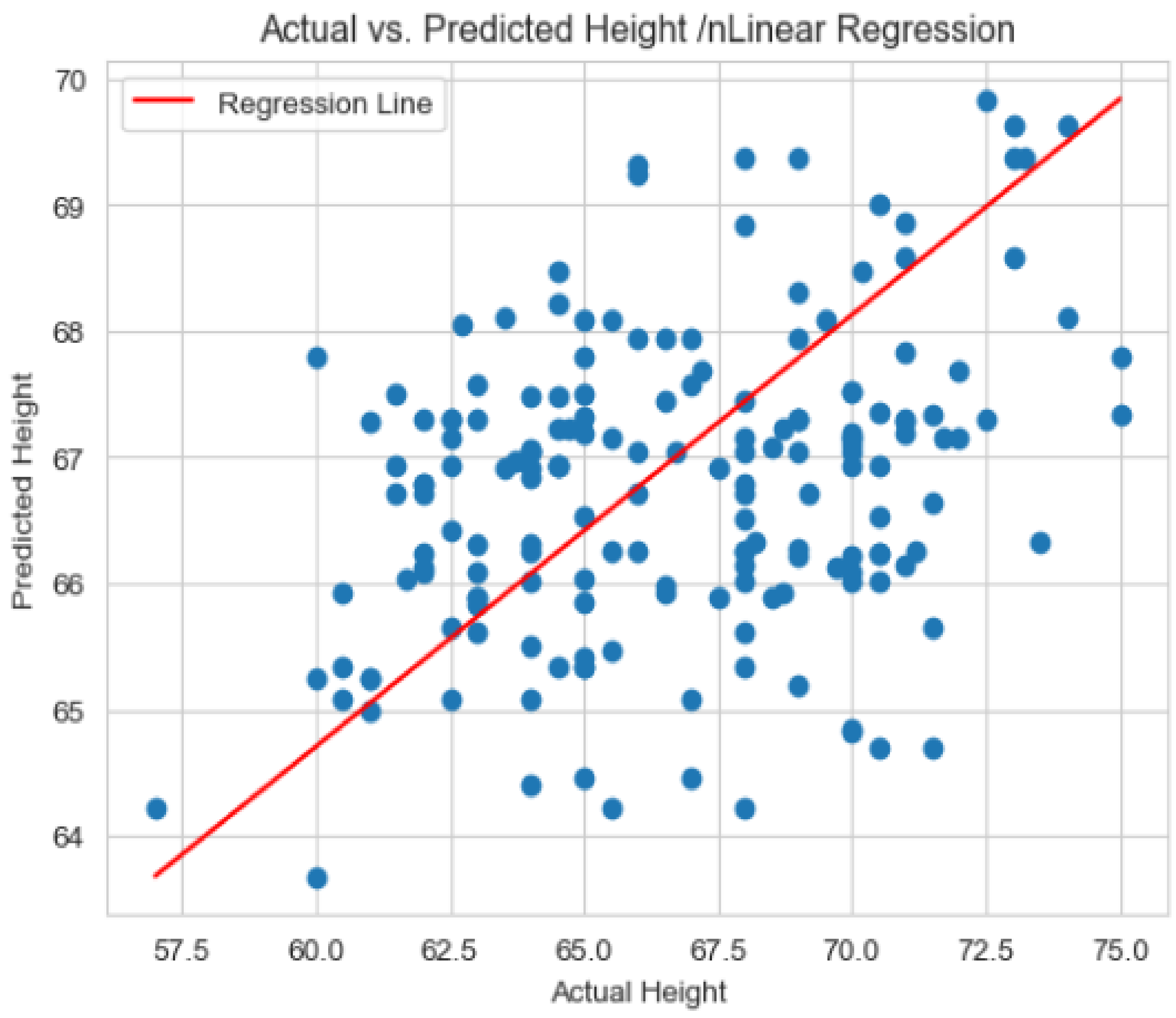
```
In [26]:  plt.figure(figsize = (12, 7))
           sns.boxplot(x='gender',y = 'height', data = df, palette = 'winter')

Out[26]: <AxesSubplot:xlabel='gender', ylabel='height'>
```

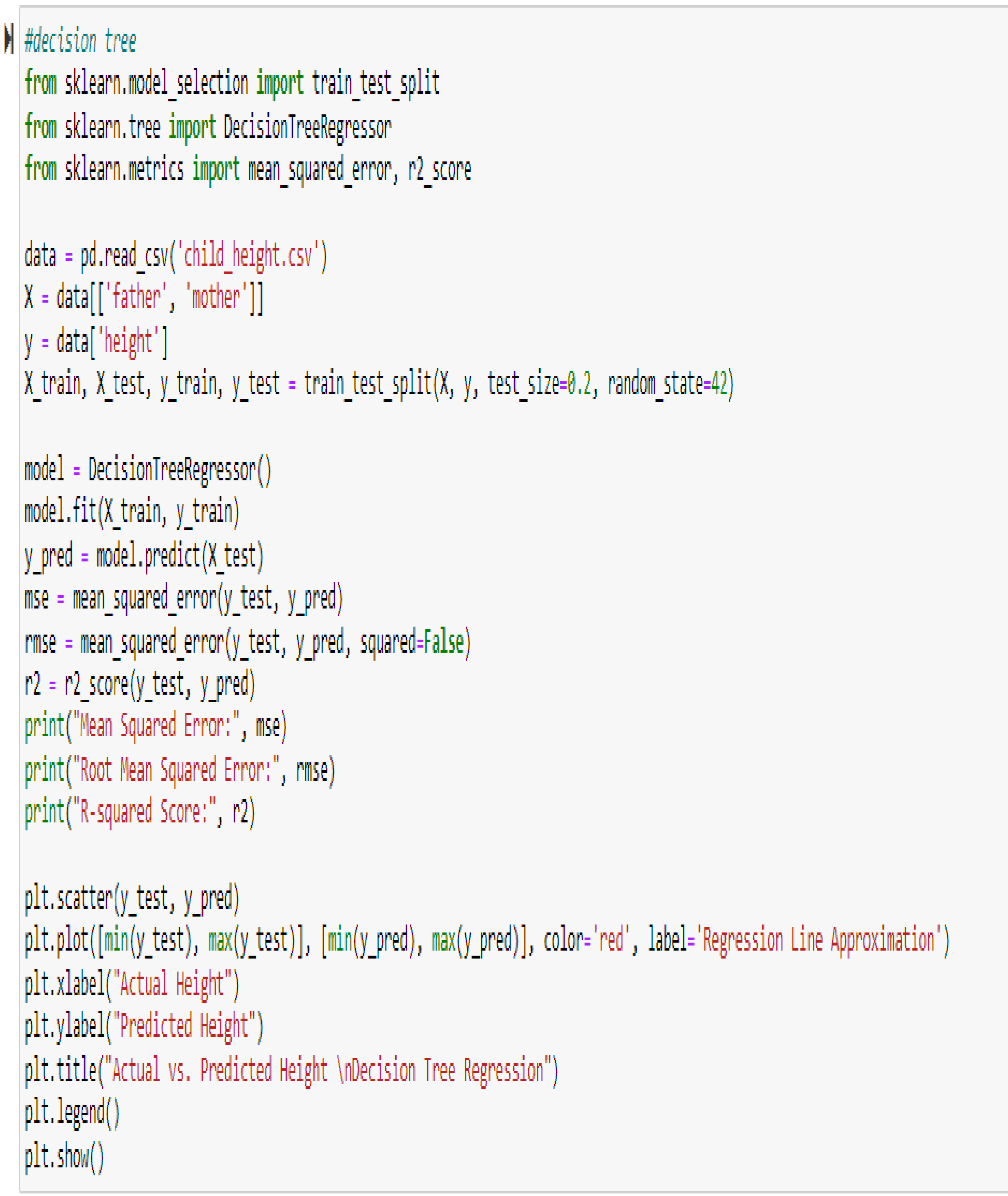




```python
#linear regression

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('child_height.csv')
X = data[['father', 'mother']]
y = data['height']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared Score:", r2)

coef = model.coef_
intercept = model.intercept_
equation = "y = {:.2f}x + {:.2f}".format(coef[0], intercept)
print("Regression Line Equation:", equation)

plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_pred), max(y_pred)], color='red', label='Regression Line')
plt.xlabel("Actual Height")
plt.ylabel("Predicted Height")
plt.title("Actual vs. Predicted Height /nLinear Regression")
plt.legend()
plt.show()
```

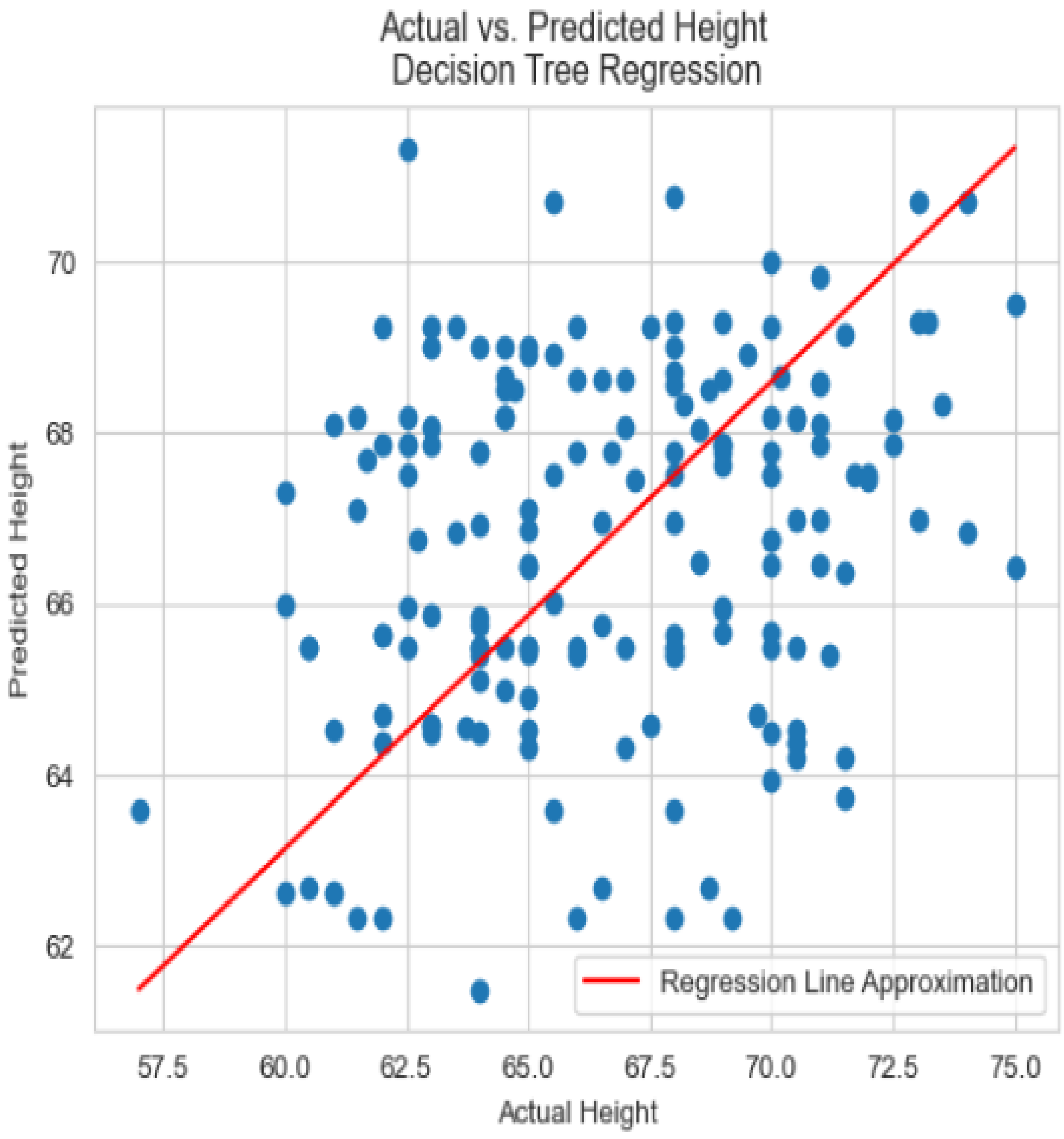
```
Mean Squared Error: 12.01999646629591
Root Mean Squared Error: 3.4669866550501616
R-squared Score: 0.11584237747872972
Regression Line Equation: y = 0.39x + 23.39
```

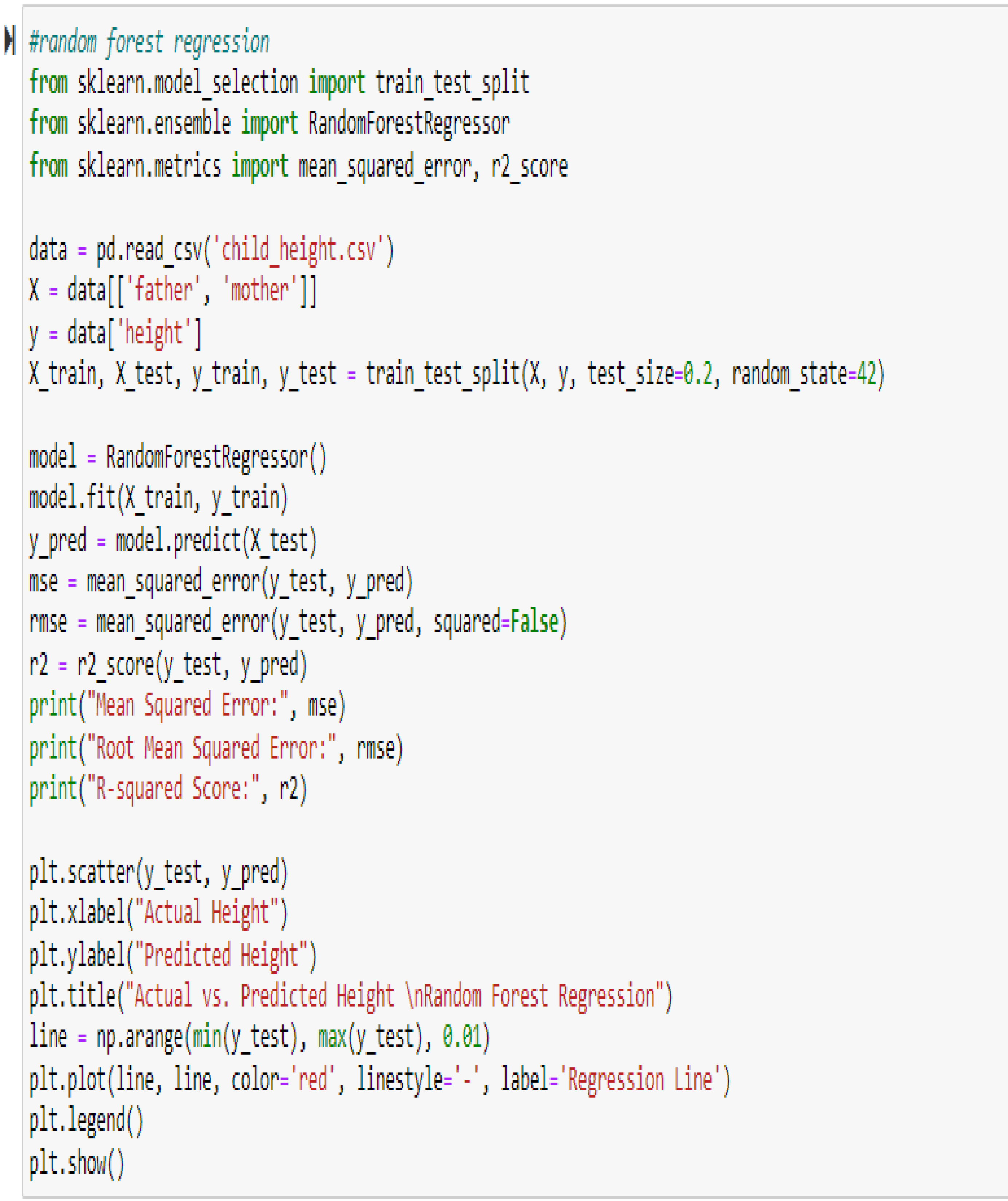




```
#decision tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('child_height.csv')
X = data[['father', 'mother']]
y = data['height']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = DecisionTreeRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared Score:", r2)

plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_pred), max(y_pred)], color='red', label='Regression Line Approximation')
plt.xlabel("Actual Height")
plt.ylabel("Predicted Height")
plt.title("Actual vs. Predicted Height \nDecision Tree Regression")
plt.legend()
plt.show()
```

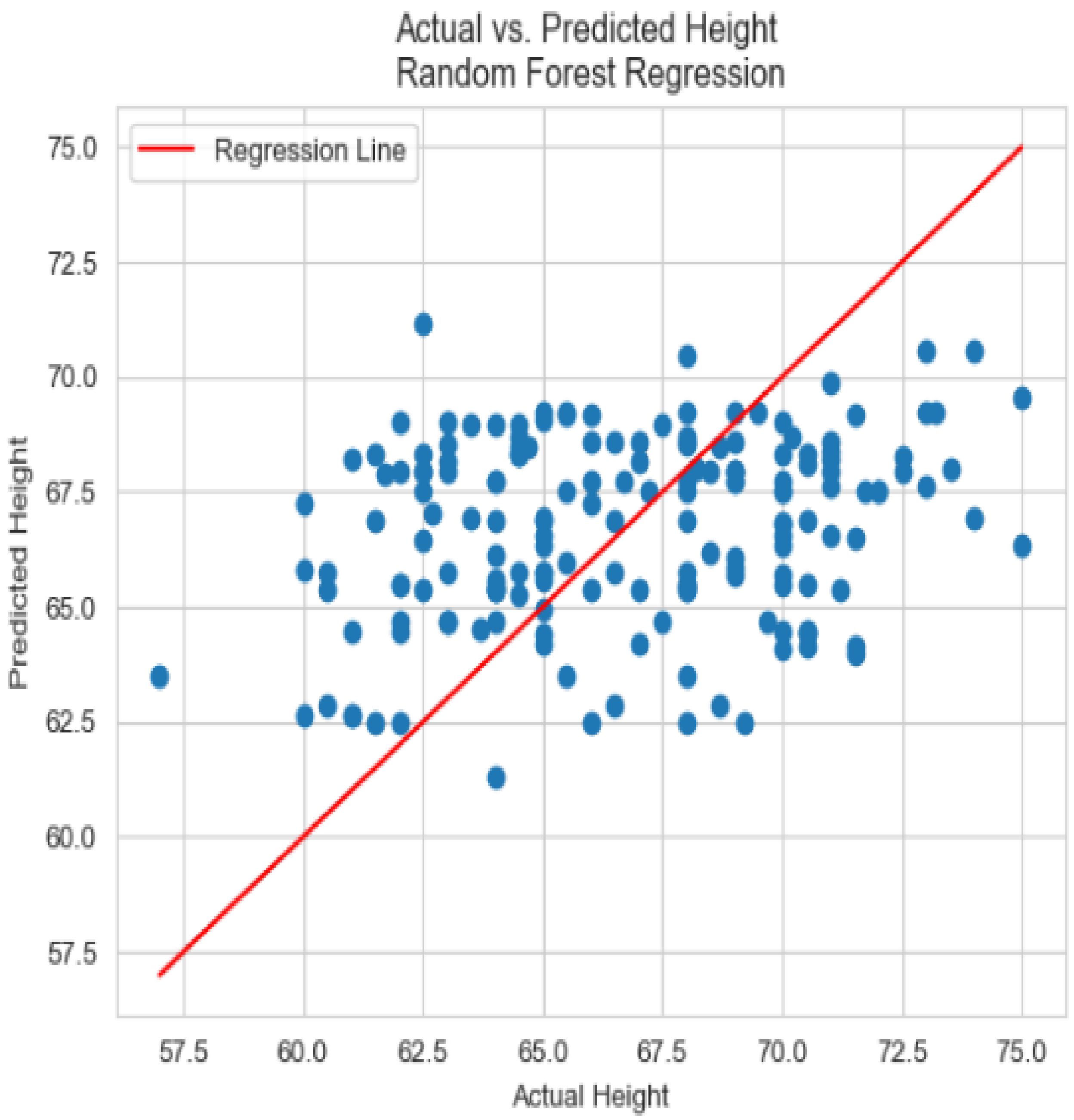
```
Mean Squared Error: 14.026768855957974
Root Mean Squared Error: 3.7452328173236404
R-squared Score: -0.03177023704741333
```

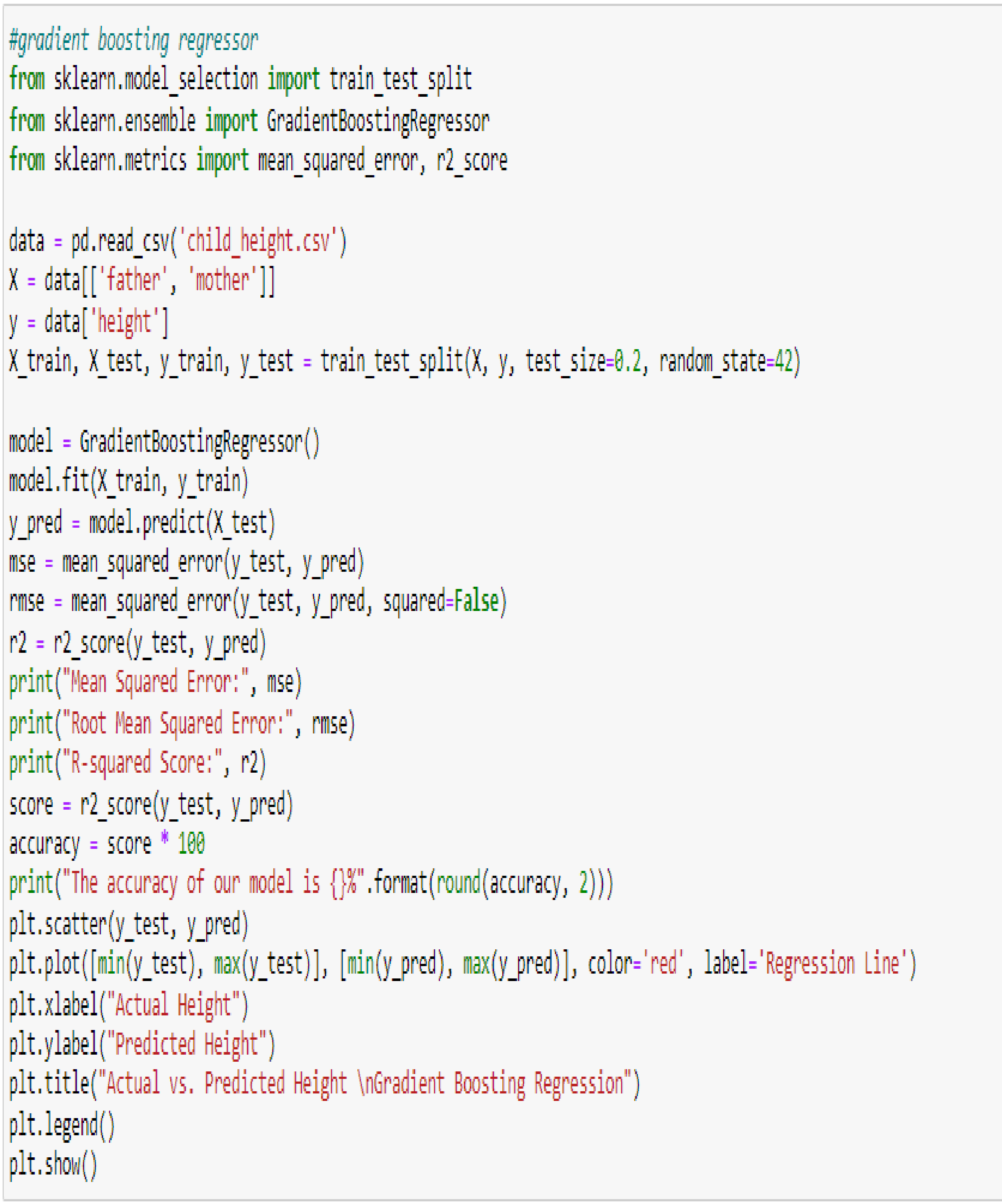




```
#random forest regression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('child_height.csv')
X = data[['father', 'mother']]
y = data['height']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared Score:", r2)

plt.scatter(y_test, y_pred)
plt.xlabel("Actual Height")
plt.ylabel("Predicted Height")
plt.title("Actual vs. Predicted Height \nRandom Forest Regression")
line = np.arange(min(y_test), max(y_test), 0.01)
plt.plot(line, line, color='red', linestyle='-', label='Regression Line')
plt.legend()
plt.show()
```

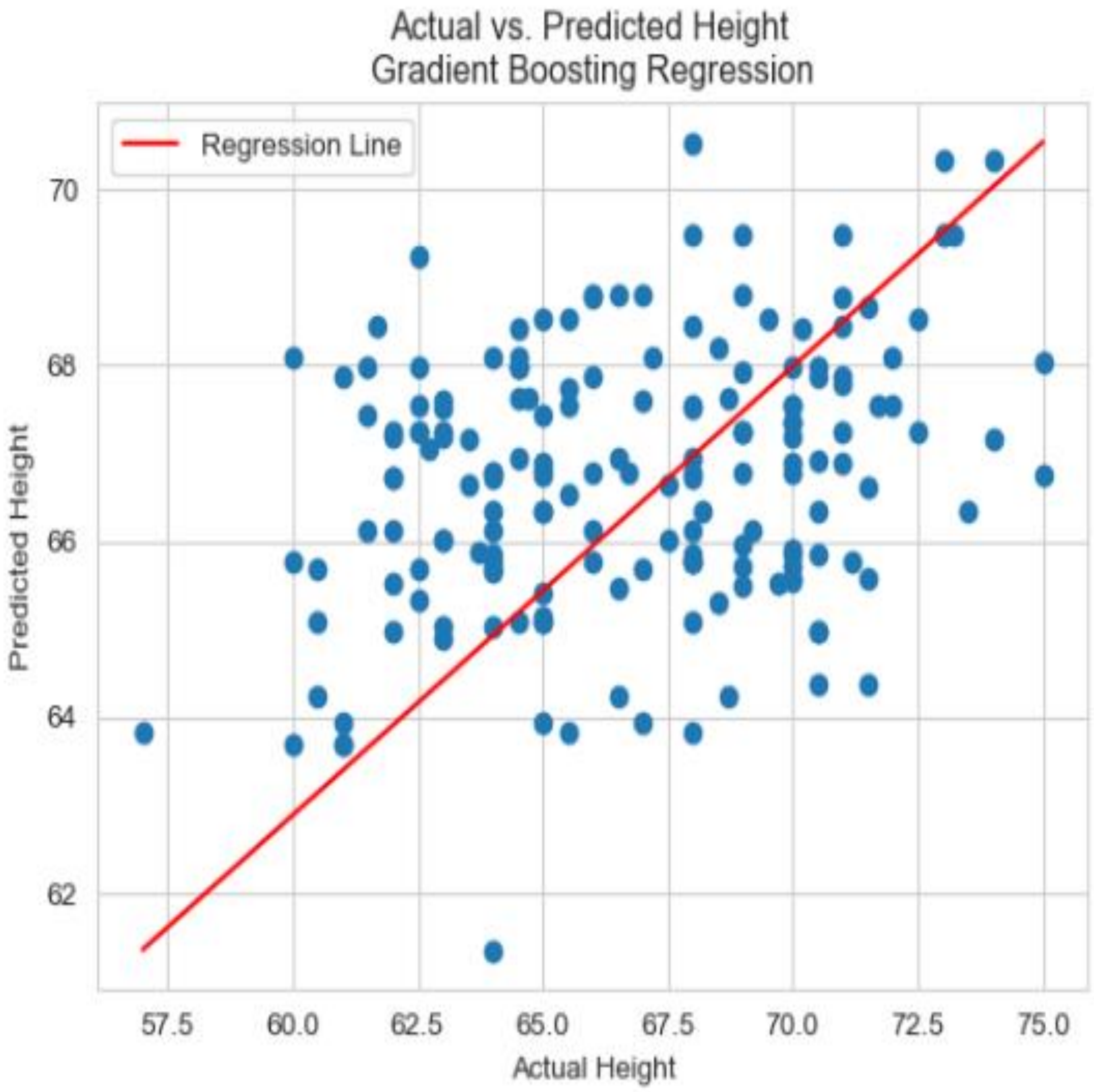
```
Mean Squared Error: 13.91096969155021
Root Mean Squared Error: 3.72974123654044
R-squared Score: -0.02325237149064674
```

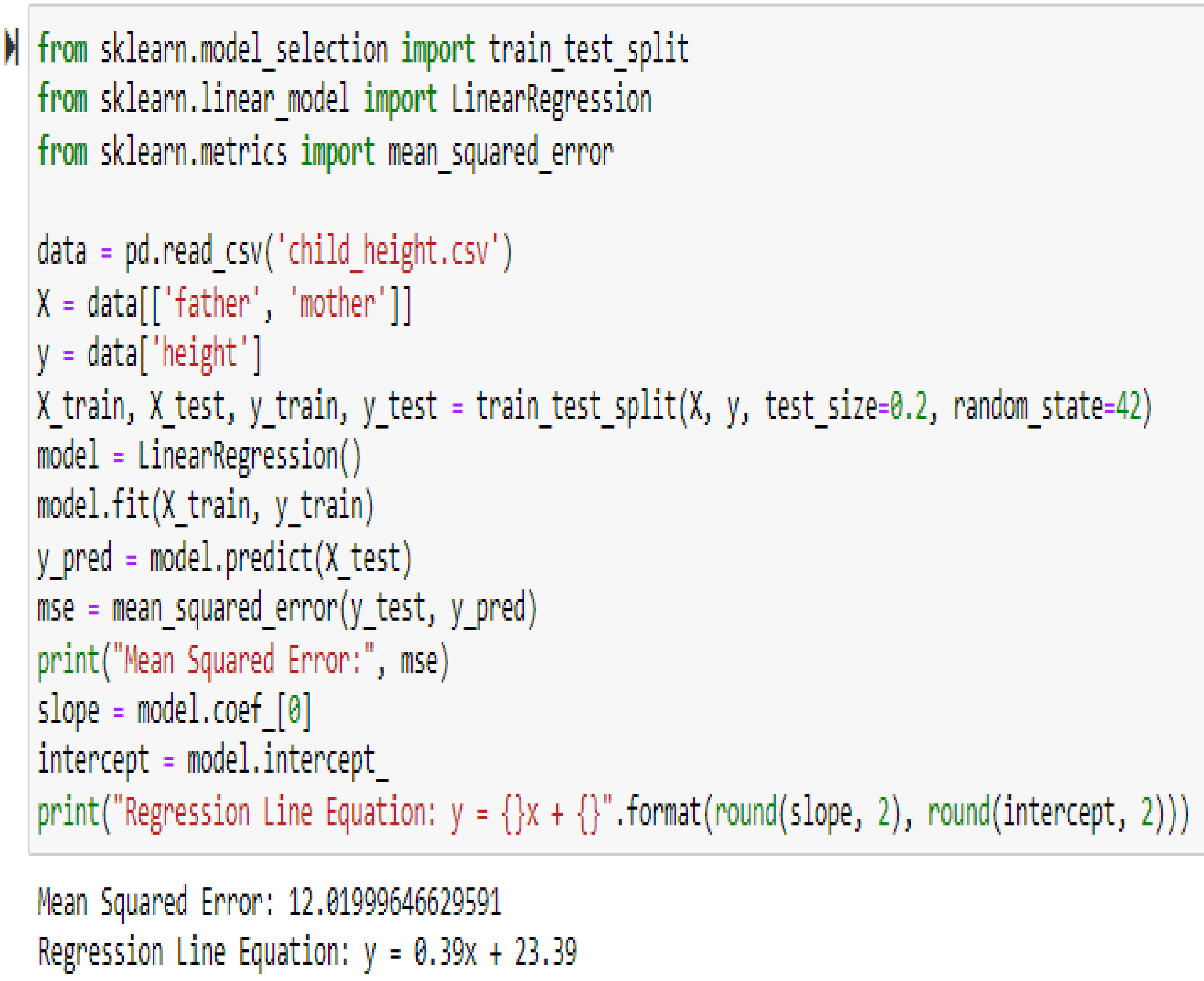




```
#gradient boosting regressor
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('child_height.csv')
X = data[['father', 'mother']]
y = data['height']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = GradientBoostingRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared Score:", r2)
score = r2_score(y_test, y_pred)
accuracy = score * 100
print("The accuracy of our model is {}%".format(round(accuracy, 2)))
plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_pred), max(y_pred)], color='red', label='Regression Line')
plt.xlabel("Actual Height")
plt.ylabel("Predicted Height")
plt.title("Actual vs. Predicted Height \nGradient Boosting Regression")
plt.legend()
plt.show()
```

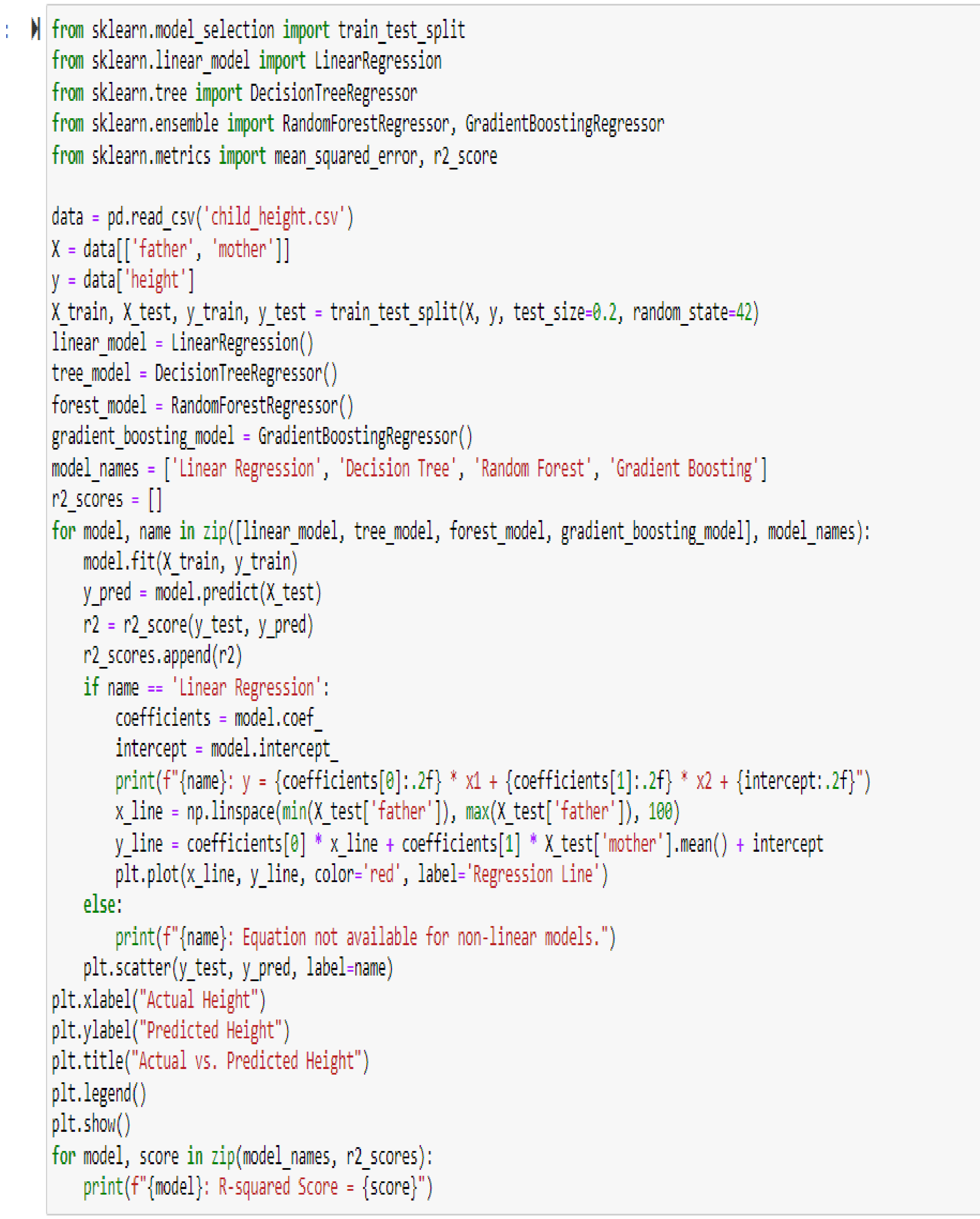
```
Mean Squared Error: 12.48694800134241
Root Mean Squared Error: 3.5336875924934863
R-squared Score: 0.08149471687691268
The accuracy of our model is 8.15%
```





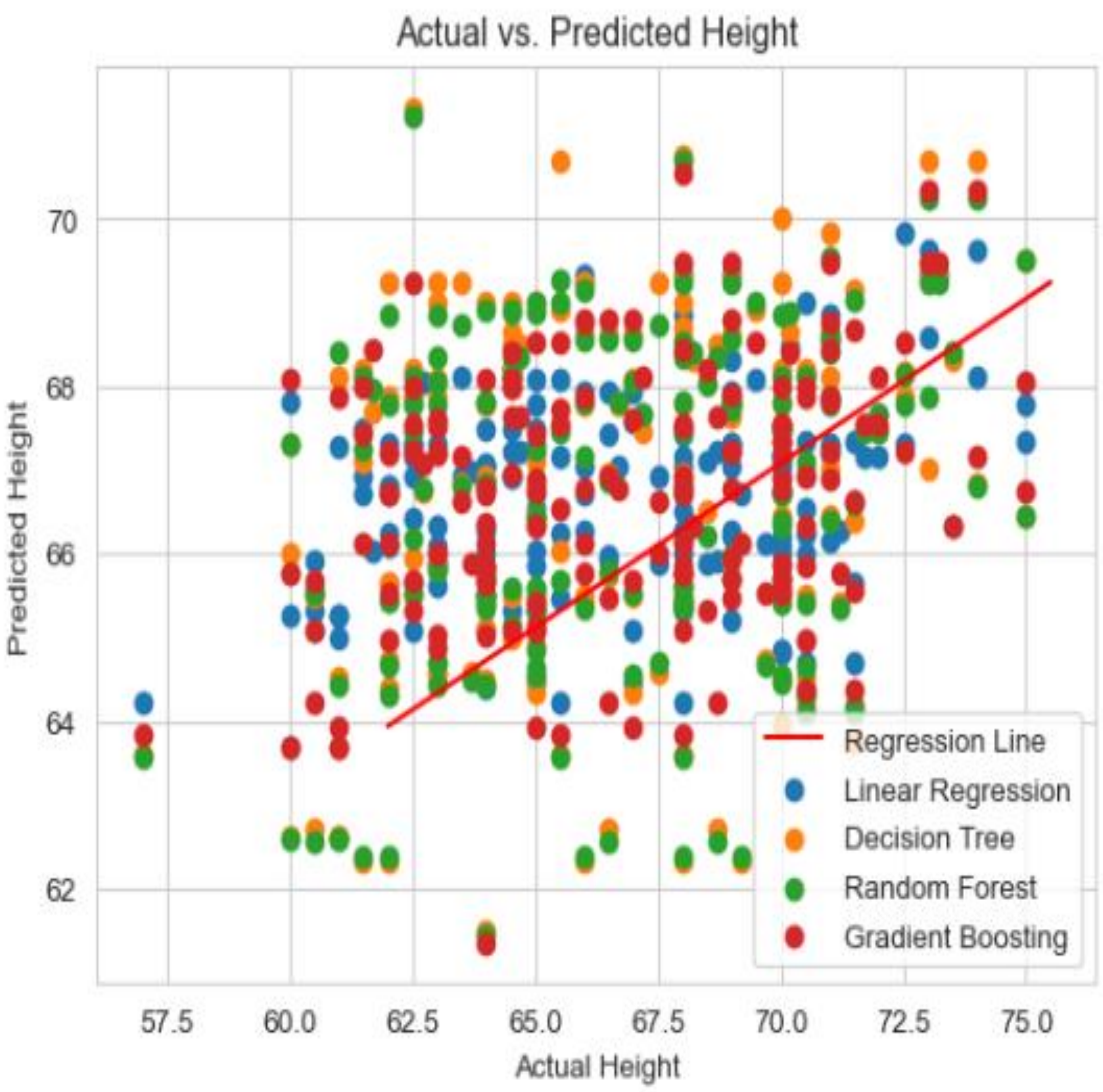
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

data = pd.read_csv('child_height.csv')
X = data[['father', 'mother']]
y = data['height']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
slope = model.coef_[0]
intercept = model.intercept_
print("Regression Line Equation: y = {}x + {}".format(round(slope, 2), round(intercept, 2)))
```

```
Mean Squared Error: 12.01999646629591
Regression Line Equation: y = 0.39x + 23.39
```



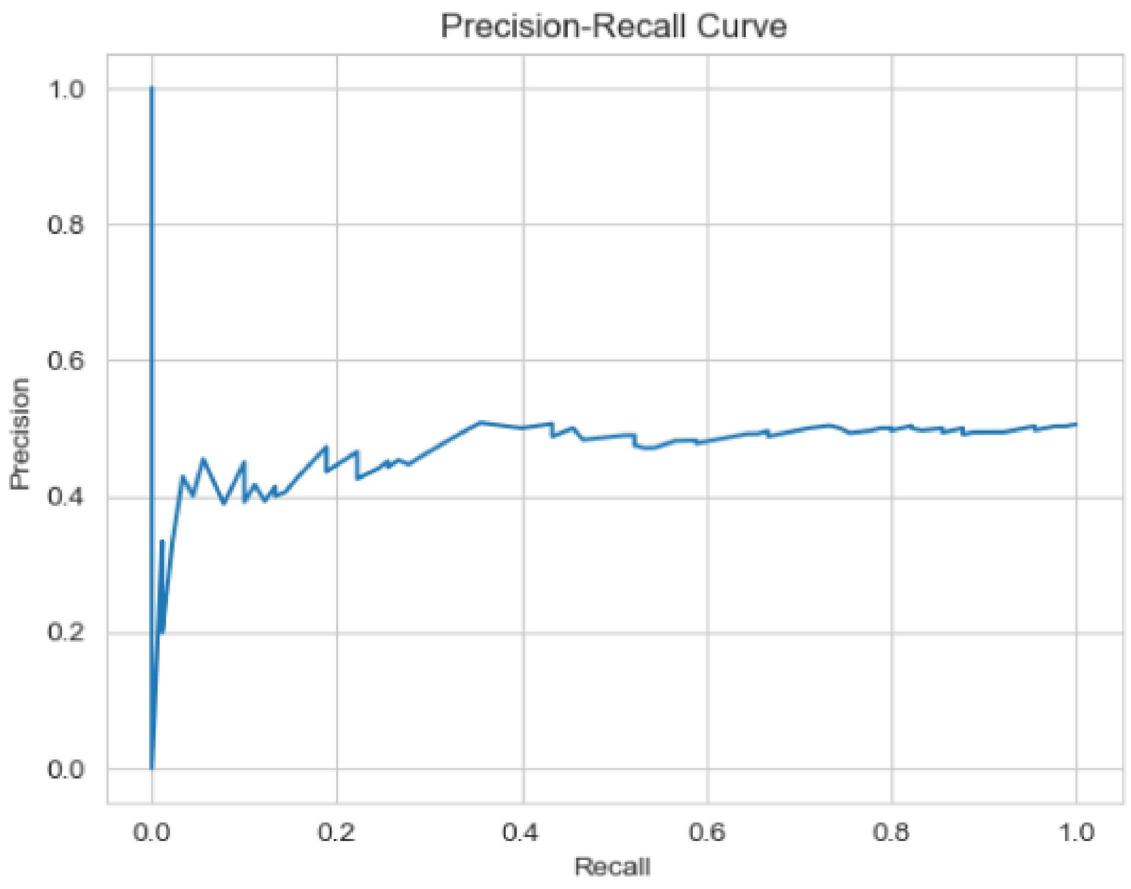
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('child_height.csv')
X = data[['father', 'mother']]
y = data['height']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
linear_model = LinearRegression()
tree_model = DecisionTreeRegressor()
forest_model = RandomForestRegressor()
gradient_boosting_model = GradientBoostingRegressor()
model_names = ['Linear Regression', 'Decision Tree', 'Random Forest', 'Gradient Boosting']
r2_scores = []
for model, name in zip([linear_model, tree_model, forest_model, gradient_boosting_model], model_names):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    r2_scores.append(r2)
    if name == 'Linear Regression':
        coefficients = model.coef_
        intercept = model.intercept_
        print(f"{name}: y = {coefficients[0]:.2f} * x1 + {coefficients[1]:.2f} * x2 + {intercept:.2f}")
        x_line = np.linspace(min(X_test['father']), max(X_test['father']), 100)
        y_line = coefficients[0] * x_line + coefficients[1] * X_test['mother'].mean() + intercept
        plt.plot(x_line, y_line, color='red', label='Regression Line')
    else:
        print(f"{name}: Equation not available for non-linear models.")
    plt.scatter(y_test, y_pred, label=name)
plt.xlabel("Actual Height")
plt.ylabel("Predicted Height")
plt.title("Actual vs. Predicted Height")
plt.legend()
plt.show()
for model, score in zip(model_names, r2_scores):
    print(f"{model}: R-squared Score = {score}")
```

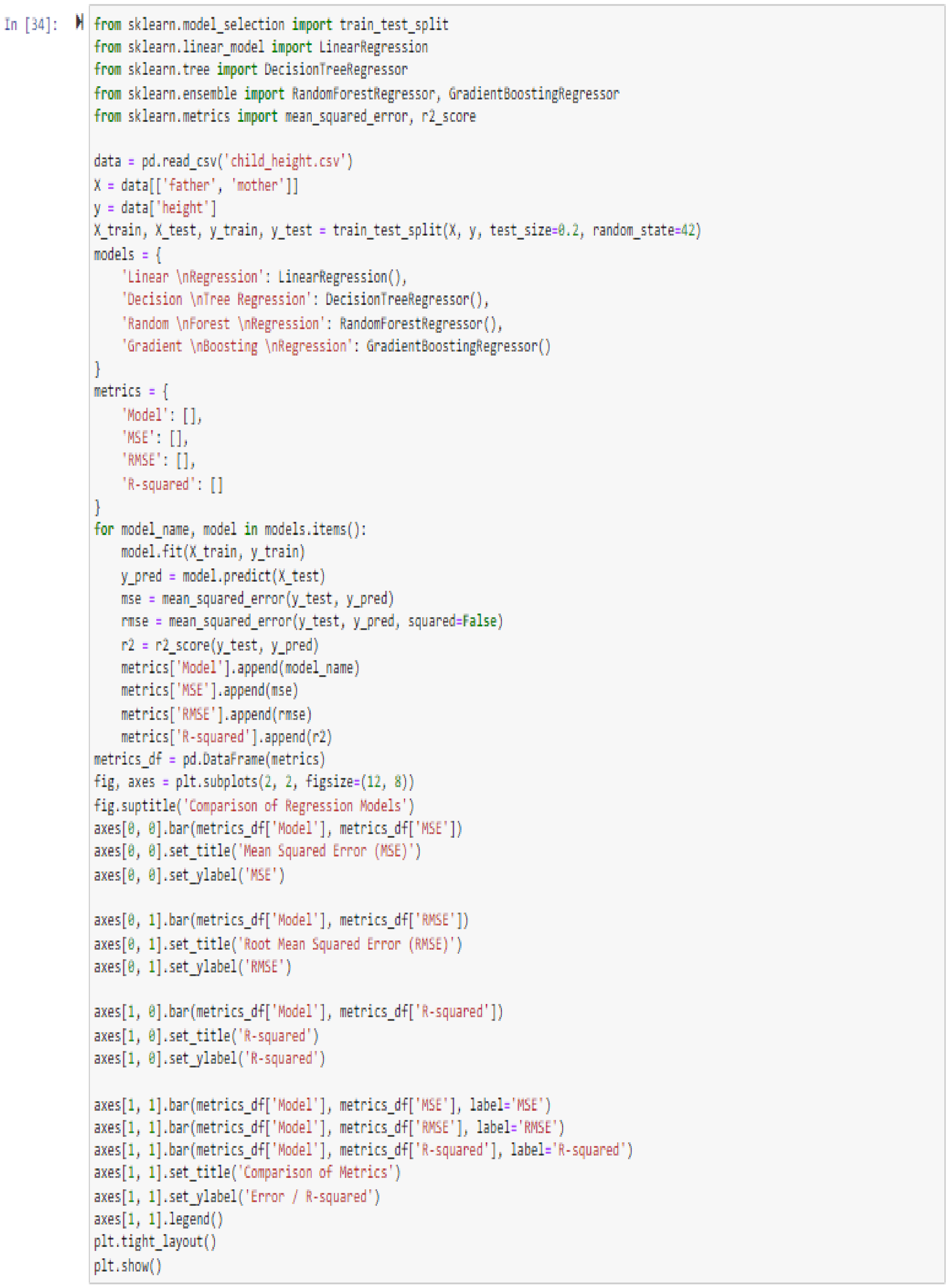
```
Linear Regression: y = 0.39 * x1 + 0.25 * x2 + 23.39
Decision Tree: Equation not available for non-linear models.
Random Forest: Equation not available for non-linear models.
Gradient Boosting: Equation not available for non-linear models.
```



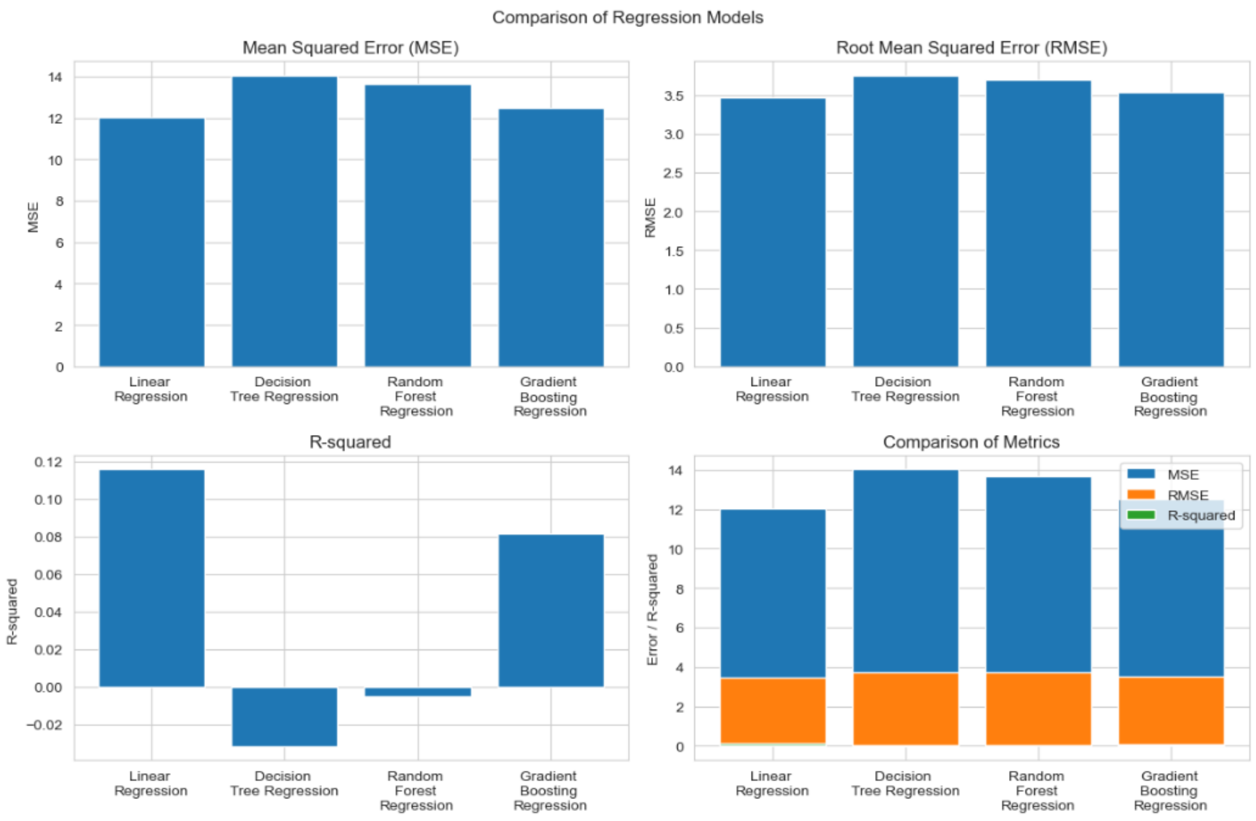
```
Linear Regression: R-squared Score = 0.11584237747872972
Decision Tree: R-squared Score = -0.03177023704741355
Random Forest: R-squared Score = -0.012100252799669198
Gradient Boosting: R-squared Score = 0.08149471687691268
```

```
In [33]:  from sklearn.model_selection import train_test_split
          from sklearn.ensemble import GradientBoostingClassifier
          from sklearn.metrics import precision_recall_curve
          from sklearn.preprocessing import LabelEncoder

          data = pd.read_csv('child_height.csv')
          X = data[['father', 'mother']]
          y = data['gender']
          le = LabelEncoder()
          y = le.fit_transform(y)
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

          model = GradientBoostingClassifier()
          model.fit(X_train, y_train)
          y_pred_proba = model.predict_proba(X_test)[:, 1]
          precision, recall, _ = precision_recall_curve(y_test, y_pred_proba)

          plt.plot(recall, precision)
          plt.xlabel('Recall')
          plt.ylabel('Precision')
          plt.title('Precision-Recall Curve')
          plt.show()
```

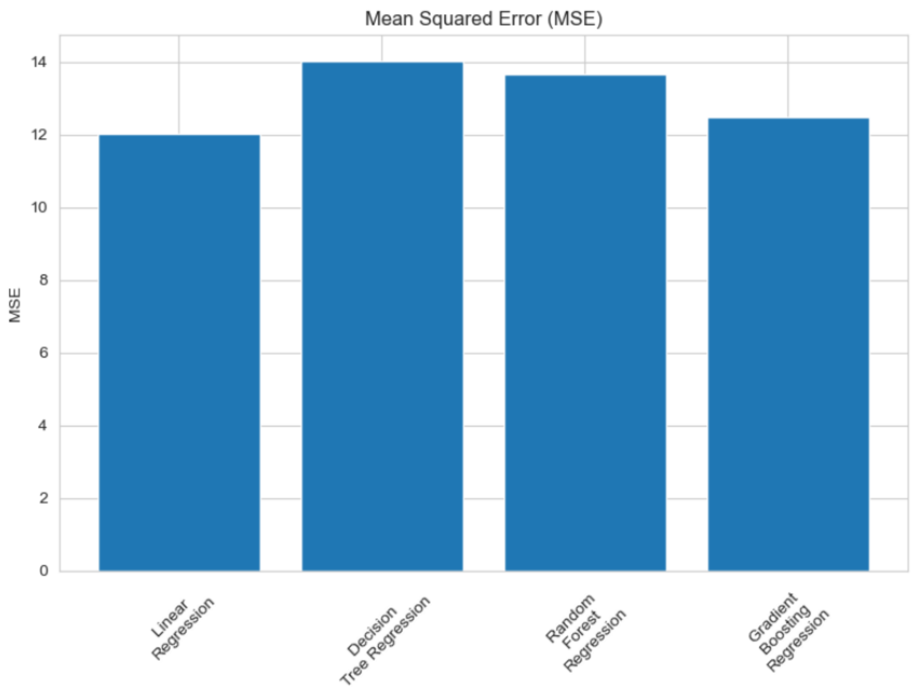




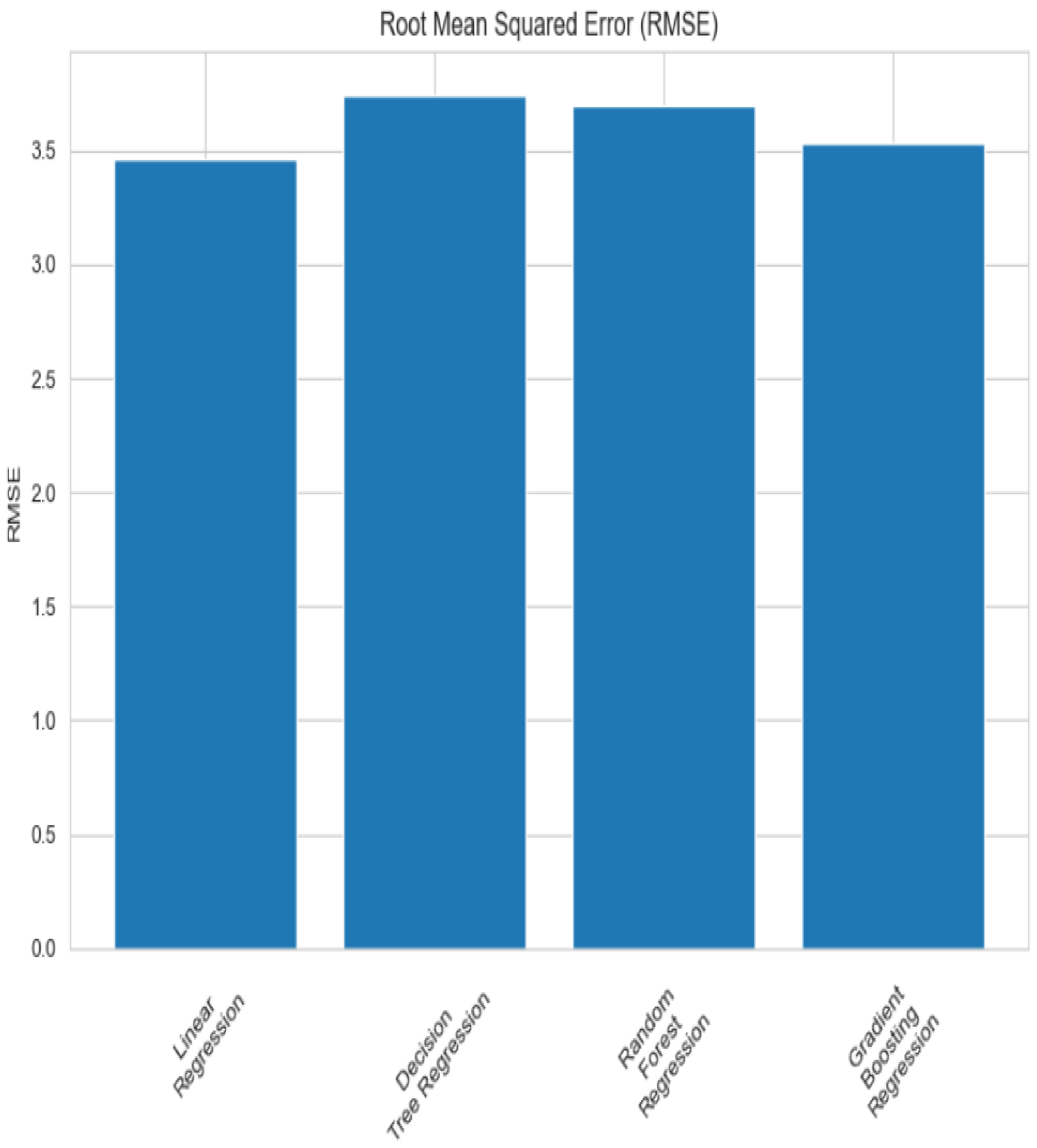
```
In [34]:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('child_height.csv')
X = data[['father', 'mother']]
y = data['height']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
models = {
    'Linear \nRegression': LinearRegression(),
    'Decision \nTree Regression': DecisionTreeRegressor(),
    'Random \nForest \nRegression': RandomForestRegressor(),
    'Gradient \nBoosting \nRegression': GradientBoostingRegressor()
}
metrics = {
    'Model': [],
    'MSE': [],
    'RMSE': [],
    'R-squared': []
}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    r2 = r2_score(y_test, y_pred)
    metrics['Model'].append(model_name)
    metrics['MSE'].append(mse)
    metrics['RMSE'].append(rmse)
    metrics['R-squared'].append(r2)
metrics_df = pd.DataFrame(metrics)
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
fig.suptitle('Comparison of Regression Models')
axes[0, 0].bar(metrics_df['Model'], metrics_df['MSE'])
axes[0, 0].set_title('Mean Squared Error (MSE)')
axes[0, 0].set_ylabel('MSE')

axes[0, 1].bar(metrics_df['Model'], metrics_df['RMSE'])
axes[0, 1].set_title('Root Mean Squared Error (RMSE)')
axes[0, 1].set_ylabel('RMSE')

axes[1, 0].bar(metrics_df['Model'], metrics_df['R-squared'])
axes[1, 0].set_title('R-squared')
axes[1, 0].set_ylabel('R-squared')

axes[1, 1].bar(metrics_df['Model'], metrics_df['MSE'], label='MSE')
axes[1, 1].bar(metrics_df['Model'], metrics_df['RMSE'], label='RMSE')
axes[1, 1].bar(metrics_df['Model'], metrics_df['R-squared'], label='R-squared')
axes[1, 1].set_title('Comparison of Metrics')
axes[1, 1].set_ylabel('Error / R-squared')
axes[1, 1].legend()
plt.tight_layout()
plt.show()
```



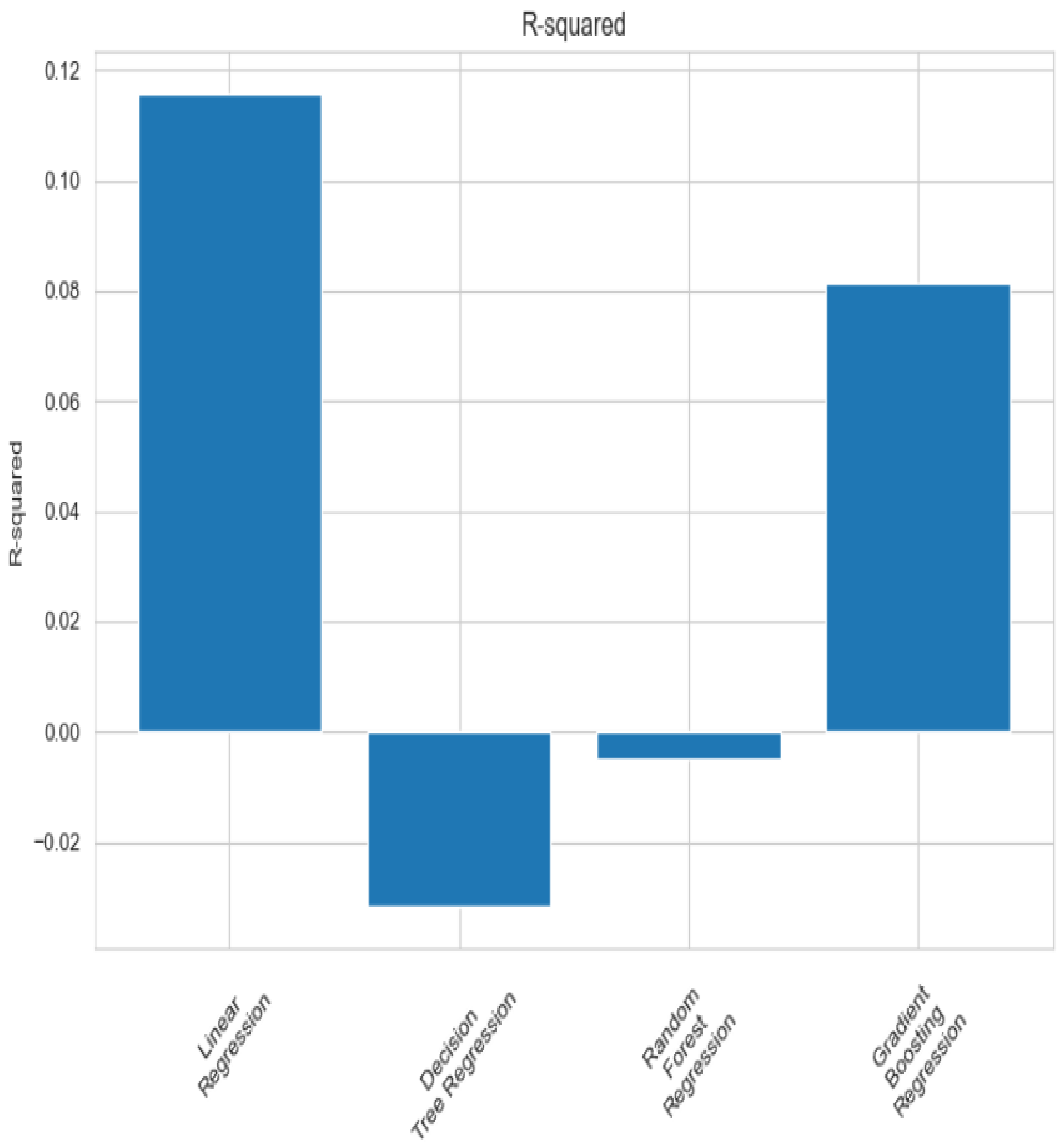
```
In [35]:  plt.figure(figsize=(8, 6))
          plt.bar(metrics_df['Model'], metrics_df['MSE'])
          plt.title('Mean Squared Error (MSE)')
          plt.ylabel('MSE')
          plt.xticks(rotation=45)
          plt.tight_layout()
          plt.show()
```



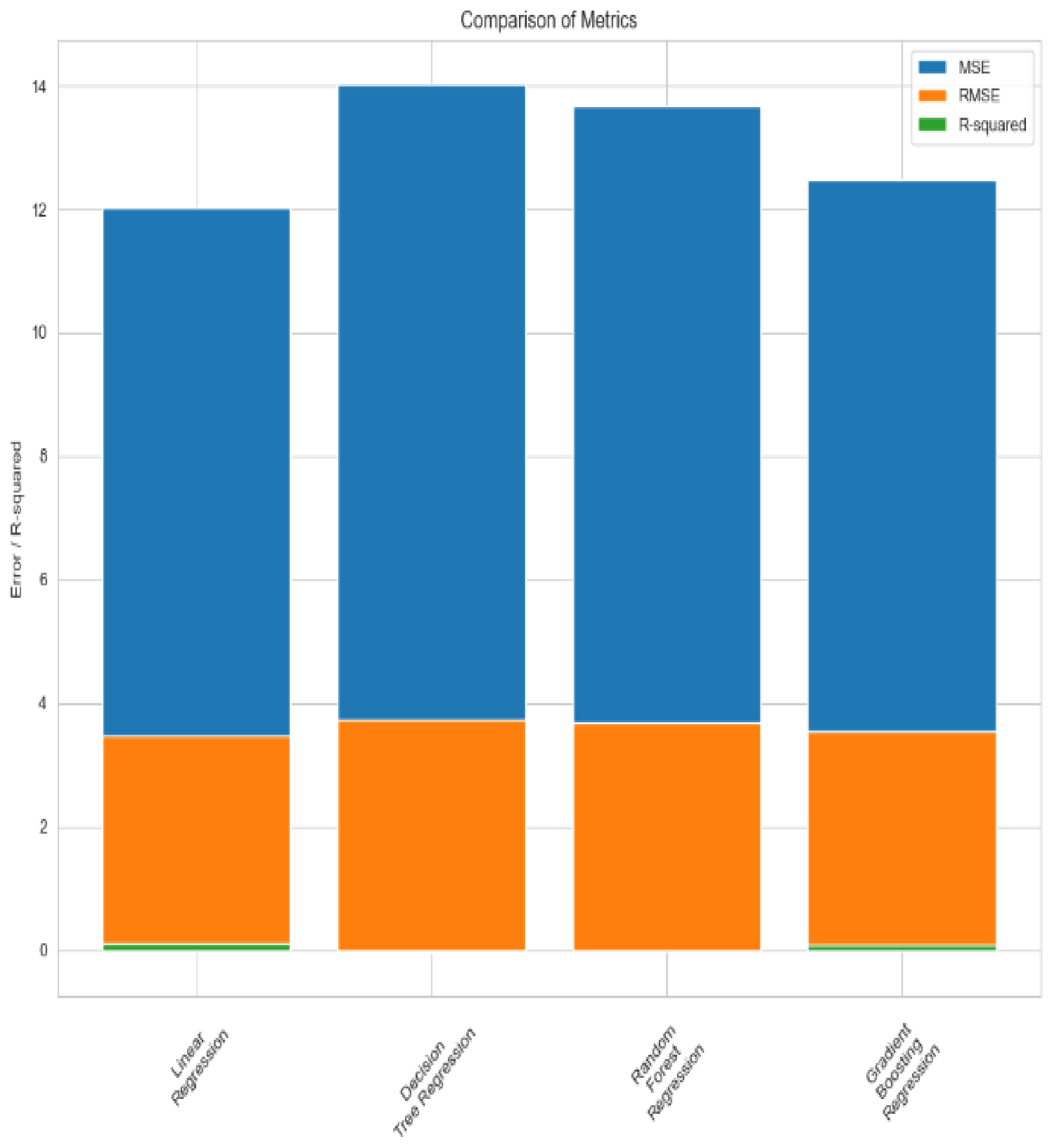
```
In [36]:  plt.figure(figsize=(8, 6))
          plt.bar(metrics_df['Model'], metrics_df['RMSE'])
          plt.title('Root Mean Squared Error (RMSE)')
          plt.ylabel('RMSE')
          plt.xticks(rotation=45)
          plt.tight_layout()
          plt.show()
```



```
In [37]:  plt.figure(figsize=(8, 6))
          plt.bar(metrics_df['Model'], metrics_df['R-squared'])
          plt.title('R-squared')
          plt.ylabel('R-squared')
          plt.xticks(rotation=45)
          plt.tight_layout()
          plt.show()
```



In [38]:
```
plt.figure(figsize=(10, 8))
plt.bar(metrics_df['Model'], metrics_df['MSE'], label='MSE')
plt.bar(metrics_df['Model'], metrics_df['RMSE'], label='RMSE')
plt.bar(metrics_df['Model'], metrics_df['R-squared'], label='R-squared')
plt.title('Comparison of Metrics')
plt.ylabel('Error / R-squared')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



Future Scope of Improvement

- While parental height is a significant predictor, there are other factors that can influence a c hild's height. Including additional variables such as ethnicity, nutrition, physical activity, and medical history may improve the accuracy of predictions.

- Employing more advanced machine learning algorithms, such as deep learning or ensemble methods, could enhance the accuracy of predictions. These techniques can identify complex patterns and relationships in large datasets, leading to improved height predictions.

- Collecting longitudinal data by tracking the growth of children over time can provide more a ccurate predictions. Analyzing growth patterns, growth spurts, and the effects of various fact ors at different stages of development can enhance the precision of height predictions.

- Access to large-scale datasets encompassing diverse populations and incorporating various v ariables can improve the robustness and generalizability of the prediction model. Integration of big data can help identify additional factors influencing height and refine prediction algori thms.

Certificate

This is to certify that Mr. Kaki Himanth of Lovely professional University, registration number: 12102363, has successfully completed a project on Child height prediction based on parents height using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Jayanth Reddy Udumula of Lovely professional University, registration number: 12103070, has successfully completed a project on Child height prediction based on parents height using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School