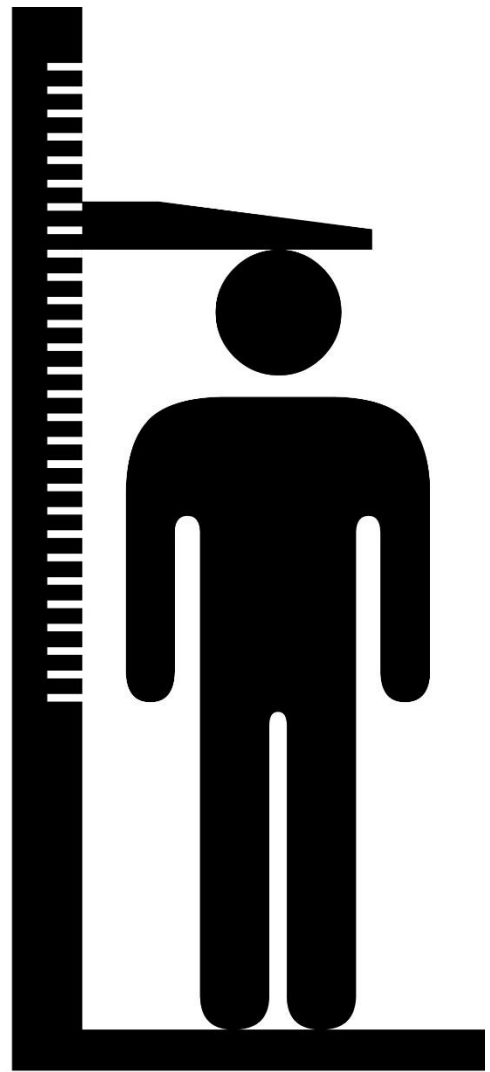
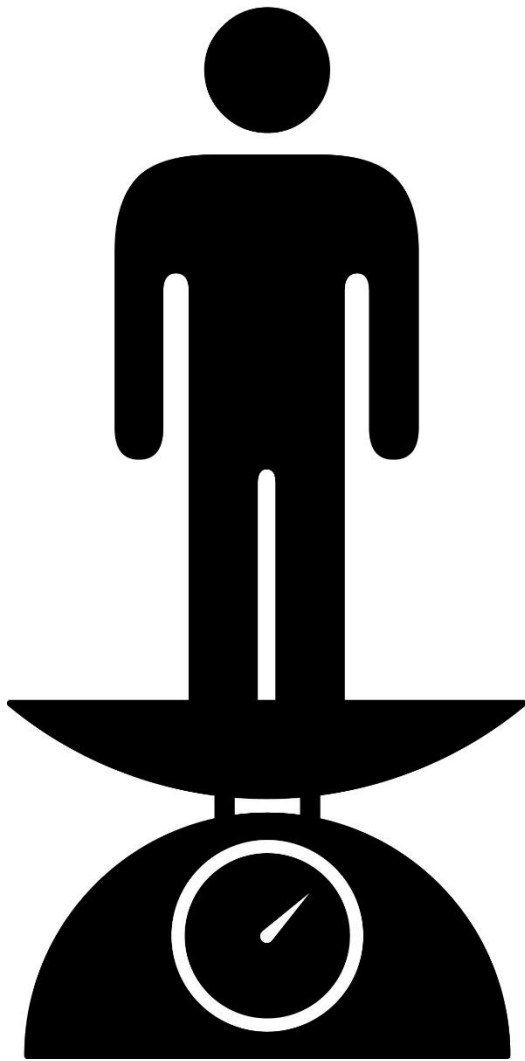


MODEL FOR PREDICTING WIEGHT USING HEIGHT



INTRODUCTION

According to most of the researchers, weight and height have a very deep connection when it comes to real life evaluation. Thus, here we make an effort to create a model which can predict weight by giving height as an input. Mostly you see that on average, men have more height than women. This theory can differ from region to region, country to country and state to state. Also, there is challenge to predict weight using height when there is no factor / feature of age, work, region etc. Thus, here we are focusing on pure statistical modeling and insights to create a relation between these two values (Height and Weight). Also, we can think in another perspective, such as what would be the gender of the candidate if his/her weight and height are given as input. Should we use some different model on that? So, what we try to achieve here is to implement multiple machine learning models on same dataset in order to create multiple insights in form of prediction which can be display in form of values or graphs.

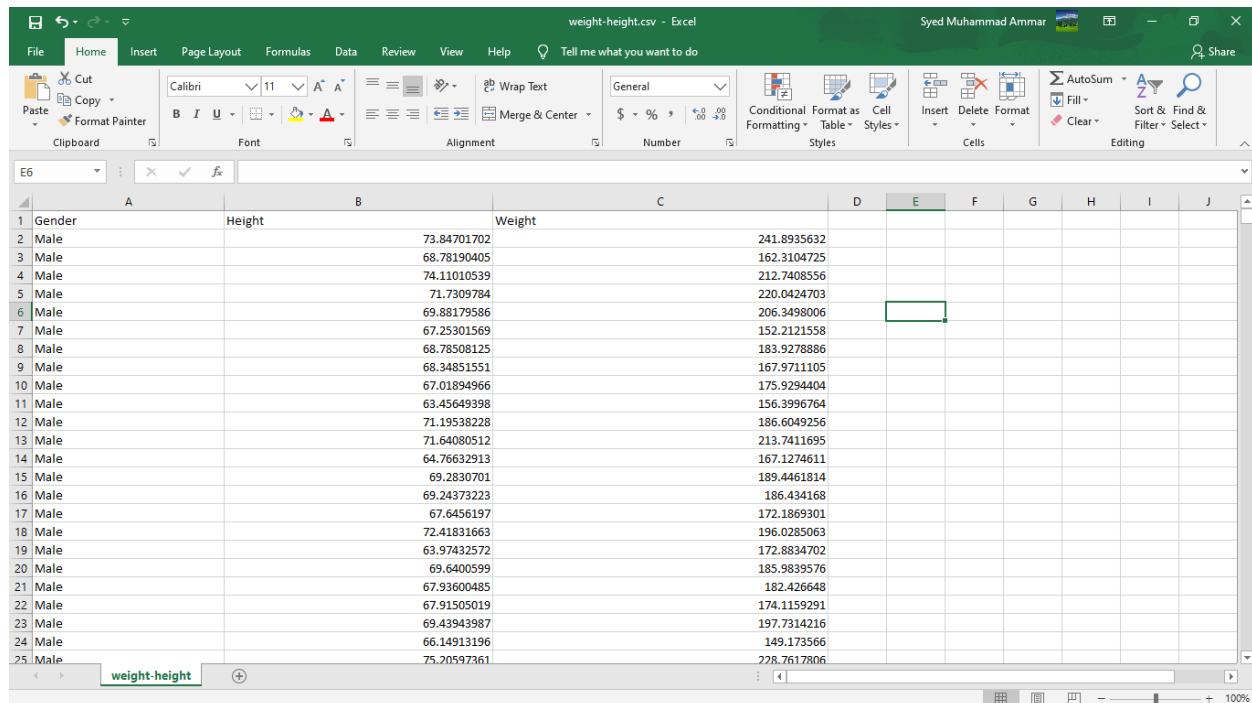
PROBLEM STATEMENT

Following are some main insights which we need to discover from the given dataset.

- What will be the weight of the candidate given height as input only?
- What will be the weight of candidate given gender and height of candidate as input?
- Is it possible to get gender of the candidate if weight and height is given as input?
- How we create connection between all these features?

DATASET

The data is collected online. It's a csv file with the size of 419 kb which holds total 10K records.



The screenshot shows an Excel spreadsheet titled "weight-height.csv - Excel". The spreadsheet contains a table with 25 rows of data. The columns are labeled A, B, and C, corresponding to Gender, Height, and Weight respectively. The data is as follows:

| Gender | Height | Weight |
|--------|-------------|-------------|
| Male | 73.84701702 | 241.8935632 |
| Male | 68.78190405 | 162.3104725 |
| Male | 74.11010539 | 212.7408556 |
| Male | 71.7309784 | 220.0424703 |
| Male | 69.88179586 | 206.3498006 |
| Male | 67.25301569 | 152.2121558 |
| Male | 68.78508125 | 183.9278886 |
| Male | 68.34851551 | 167.9711105 |
| Male | 67.01894966 | 175.9294404 |
| Male | 63.45649398 | 156.3996764 |
| Male | 71.19538228 | 186.6049256 |
| Male | 71.64080512 | 213.7411695 |
| Male | 64.76632913 | 167.1274611 |
| Male | 69.2830701 | 189.4461814 |
| Male | 69.24373223 | 186.434168 |
| Male | 67.6456197 | 172.1869301 |
| Male | 72.41831663 | 196.0285063 |
| Male | 63.97432572 | 172.8834702 |
| Male | 69.6400599 | 185.9839576 |
| Male | 67.93600485 | 182.426648 |
| Male | 67.91505019 | 174.1159291 |
| Male | 69.43943987 | 197.7314216 |
| Male | 66.14913196 | 149.173566 |
| Male | 75.20597361 | 228.7617806 |

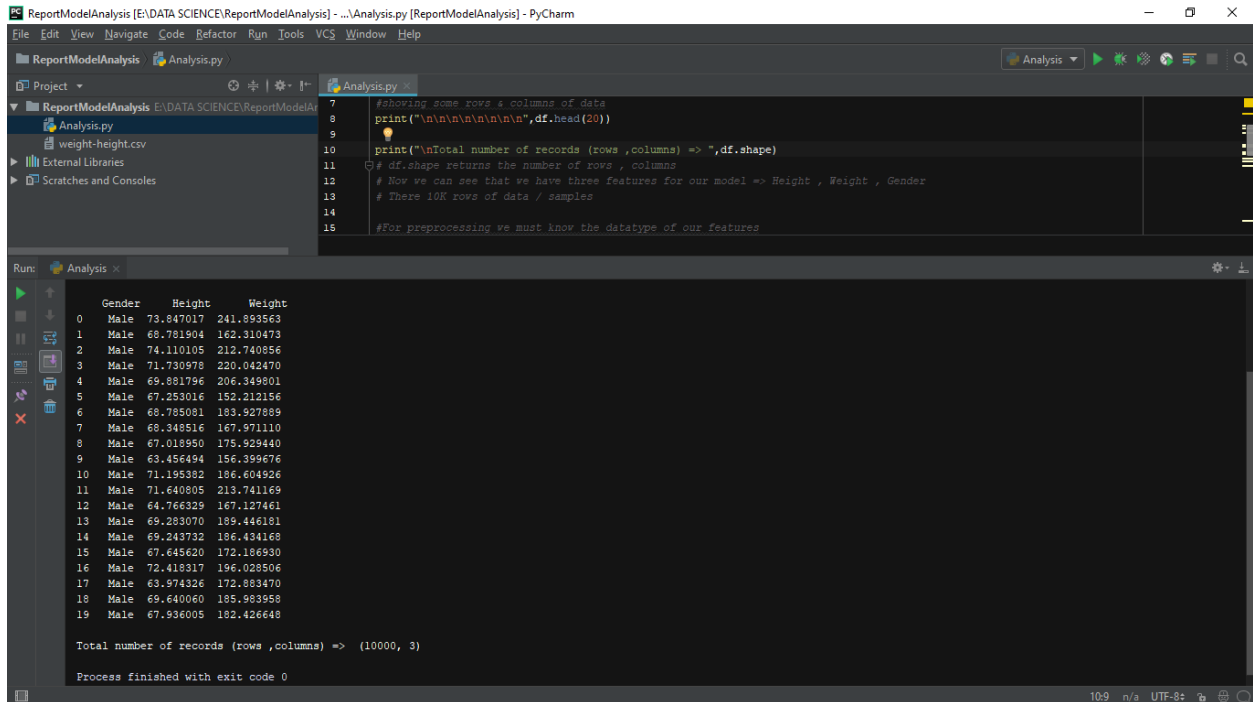
As you can see, three features are given

- Gender
- Weight (Pounds)
- Height (Inches)

Now, to predict weight, we will make weight as our target class. Now what will be our input? Should we only go for height or both (height and gender). To know which model will suit better, we have to experiment both scenarios.

Also, we need to see if data need to clean or not. For this, we will evaluate data using **pandas** library in order to preprocess our data in a right and effective manner.

Here we import our data into data-frame using pandas library



The screenshot shows a PyCharm IDE with a project named 'ReportModelAnalysis'. The file 'Analysis.py' is open, containing the following code:

```
7 #showing some rows & columns of data
8 print("\n\n\n\n\n\n\n",df.head(20))
9
10 print("\n\nTotal number of records (rows ,columns) => ",df.shape)
11 # df.shape returns the number of rows , columns
12 # Now we can see that we have three features for our model => Height , Weight , Gender
13 # There 10K rows of data / samples
14
15 #For preprocessing we must know the datatype of our features
```

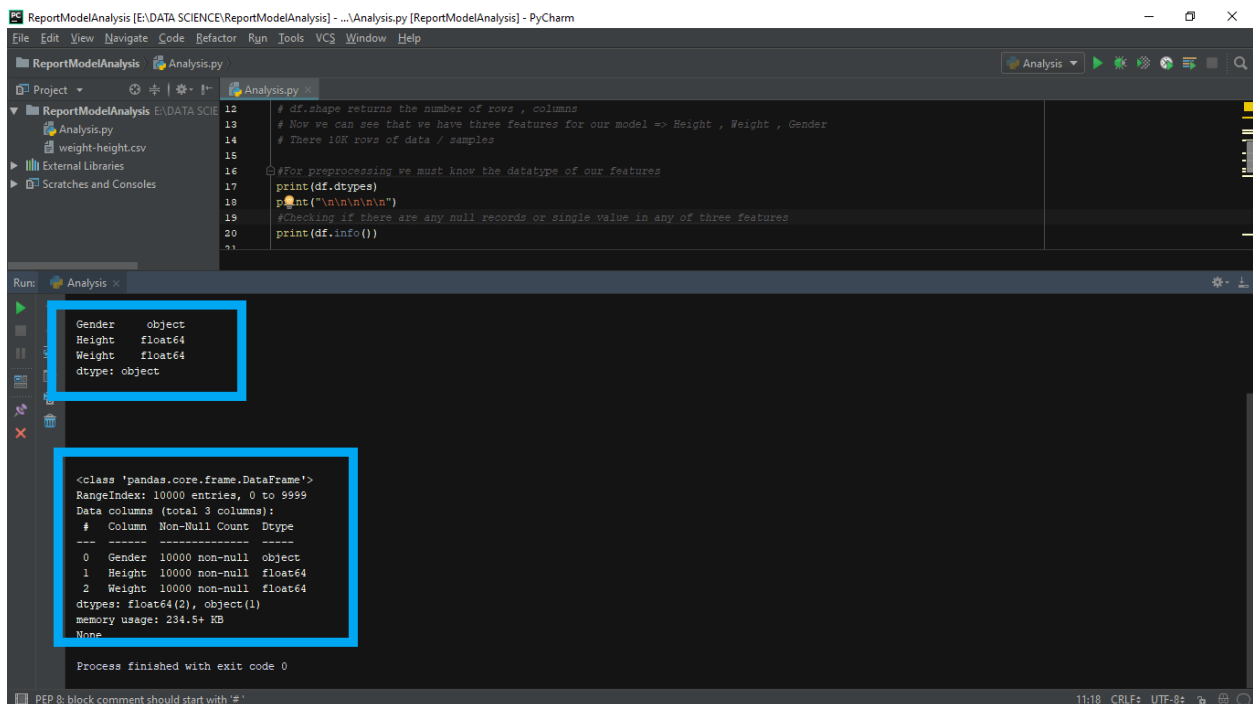
The Run window shows the output of the code:

```
Gender  Height  Weight
0  Male  73.847017  241.893563
1  Male  68.781904  152.310473
2  Male  74.110105  212.740856
3  Male  71.730978  220.042470
4  Male  69.881796  206.349801
5  Male  67.253016  152.212156
6  Male  68.785081  183.927889
7  Male  68.348516  167.971110
8  Male  67.018950  175.929440
9  Male  63.456494  156.399676
10 Male  71.195382  186.604926
11 Male  71.640805  213.741169
12 Male  64.766329  167.127461
13 Male  69.283070  189.446181
14 Male  69.243732  186.434168
15 Male  67.645620  172.186930
16 Male  72.418317  196.028506
17 Male  63.974326  172.883470
18 Male  69.640060  185.983958
19 Male  67.936005  182.426648

Total number of records (rows ,columns) => (10000, 3)

Process finished with exit code 0
```

To preprocess our data effectively, we must know the data types of every feature and we must check if there is any missing (Null) value in dataset.



The screenshot shows the same PyCharm IDE with the 'Analysis.py' file updated with the following code:

```
12 # df.shape returns the number of rows , columns
13 # Now we can see that we have three features for our model => Height , Weight , Gender
14 # There 10K rows of data / samples
15
16 #For preprocessing we must know the datatype of our features
17 print(df.dtypes)
18 print("\n\n\n\n\n\n\n")
19 #Checking if there are any null records or single value in any of three features
20 print(df.info())
21
```

The Run window shows the output of the code:

```
Gender    object
Height    float64
Weight    float64
dtype: object

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ---
 0  Gender  10000 non-null  object
 1  Height  10000 non-null  float64
 2  Weight  10000 non-null  float64
dtypes: float64(2), object(1)
memory usage: 234.5+ MB
None

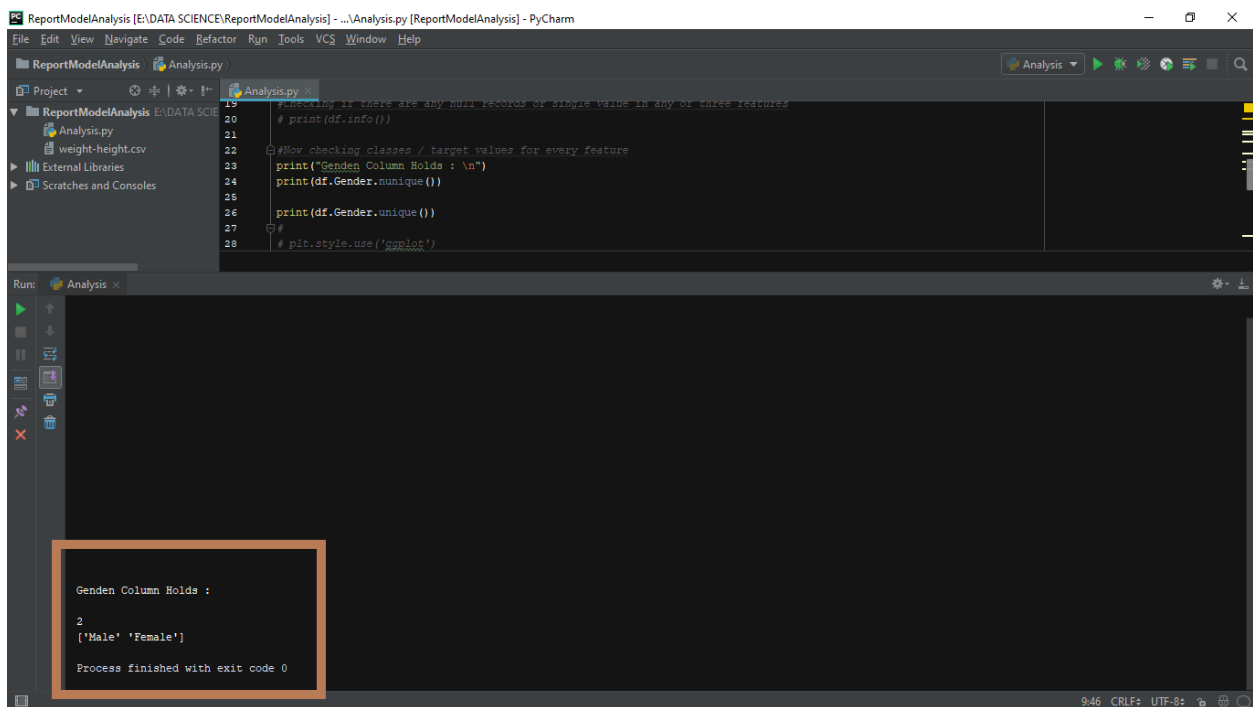
Process finished with exit code 0
```

As no null values are present among all three features, so we are in good position to go ahead. So, now its time to study the features.

1. Categorical Feature => Gender

2. Non-Categorical Features => Height, Weight

Let's talk about gender. Its important that we should now how many values in terms of uniqueness, our feature holds.



The screenshot shows the PyCharm IDE interface. The top pane displays the code in `Analysis.py`:

```
19 #Checking if there are any null records or single value in any of three features
20 # print(df.info())
21
22 #Now checking classes / target values for every feature
23 print("Genden Column Holds : \n")
24 print(df.Gender.unique())
25
26 print(df.Gender.unique())
27
28 # plt.style.use('ggplot')
```

The bottom pane shows the output of the script:

```
Genden Column Holds :
2
['Male' 'Female']
Process finished with exit code 0
```

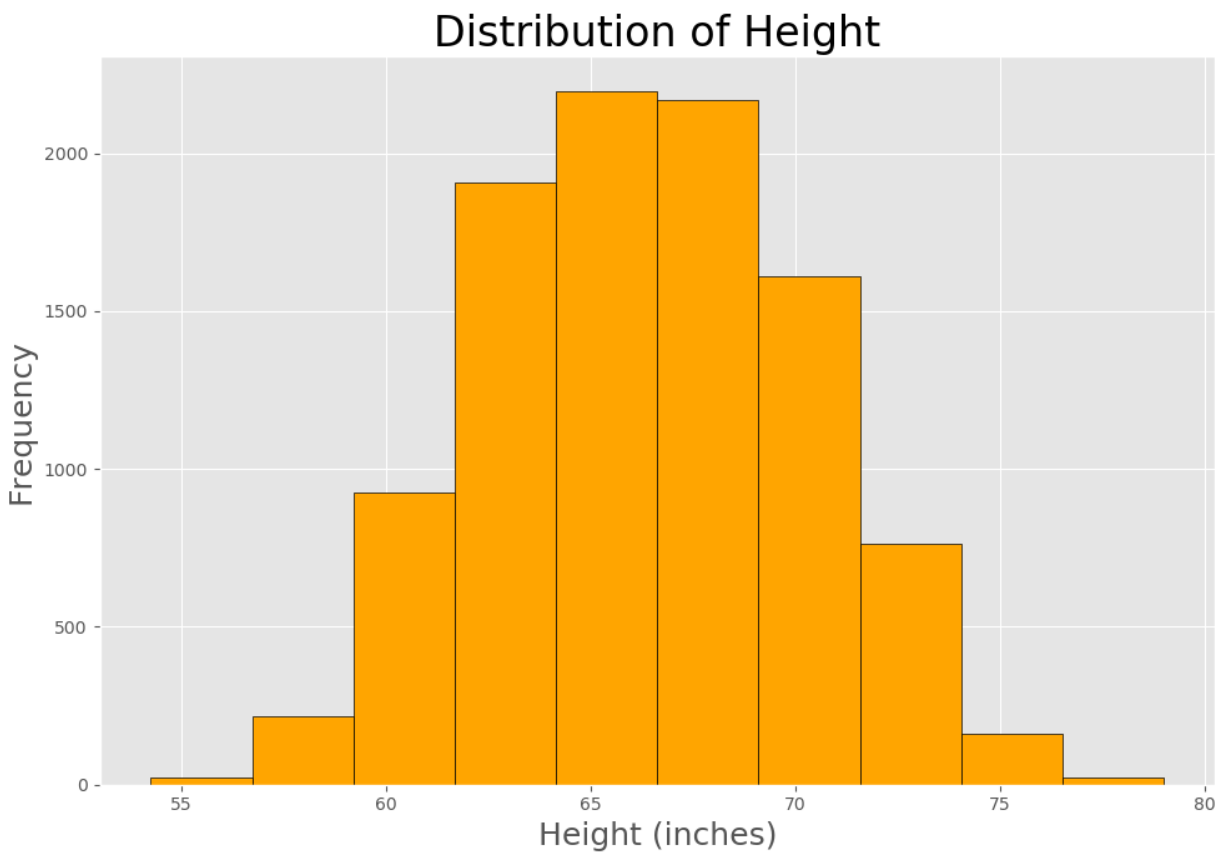
So, we see that our gender column holds two values:

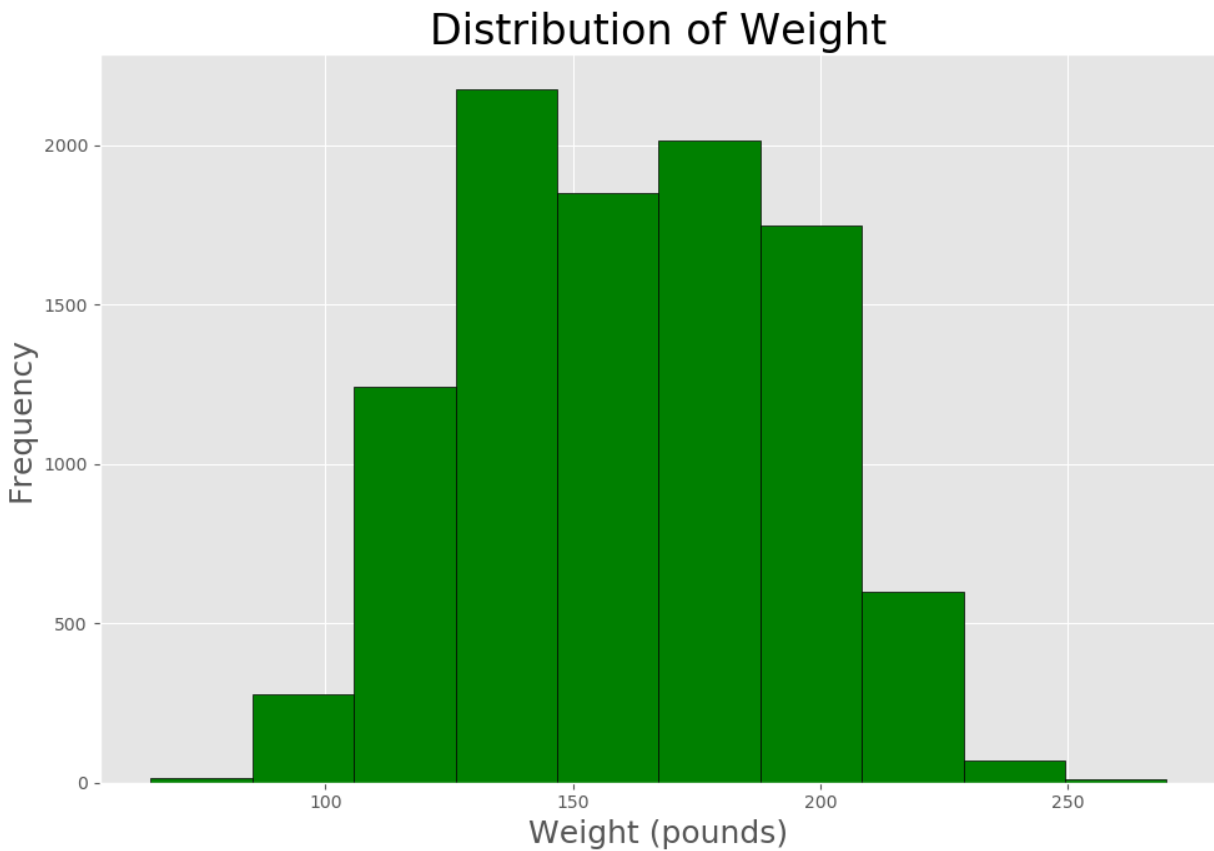
- Male
- Female

Now we must study the pattern of height and weight with respect to the gender. To better understand the distribution of the variables Height and Weight, we can

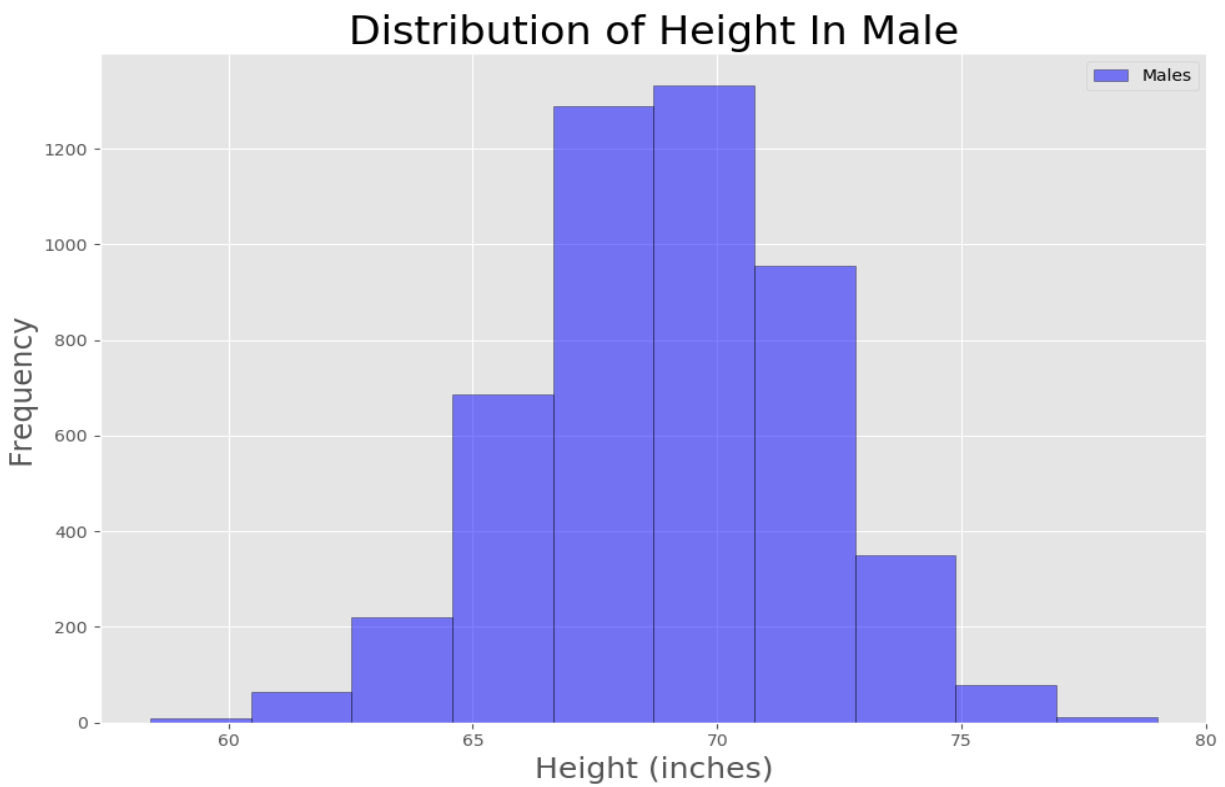
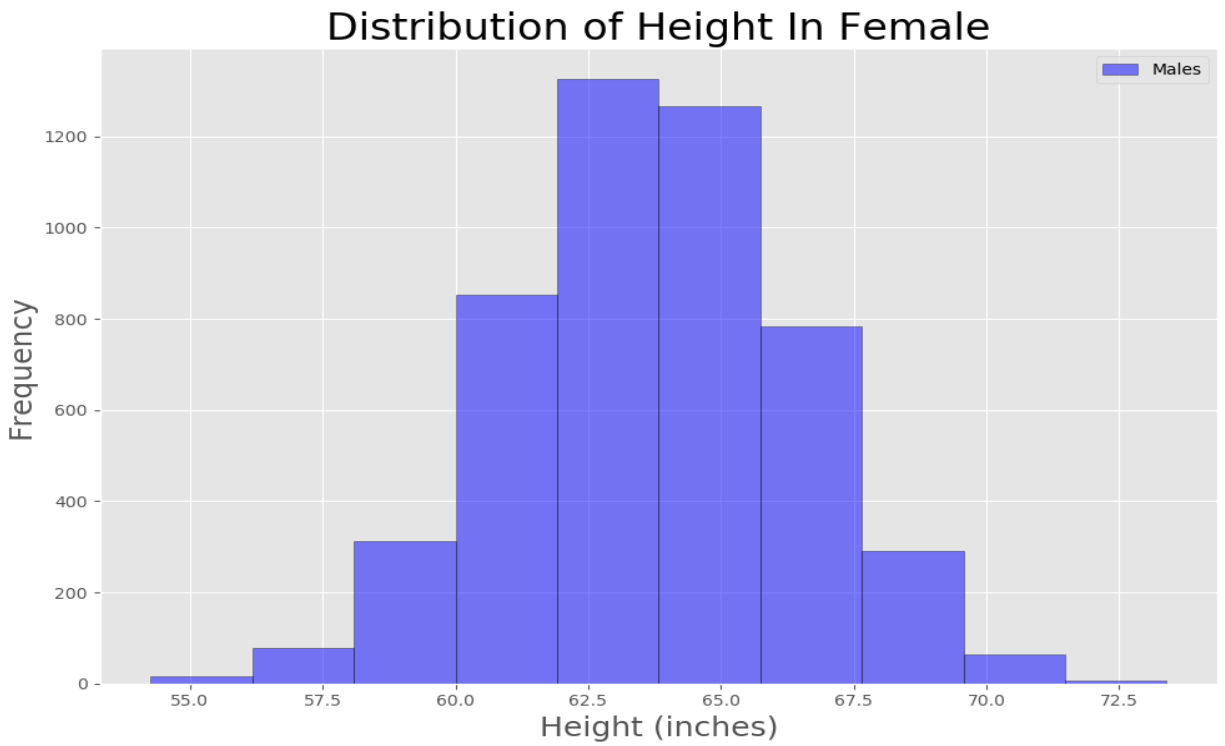
simply plot both variables using histograms. Histograms are plots that show the distribution of a numeric variable, grouping data into bins. The height of the bar represents the number of observations per bin.

These are histograms:



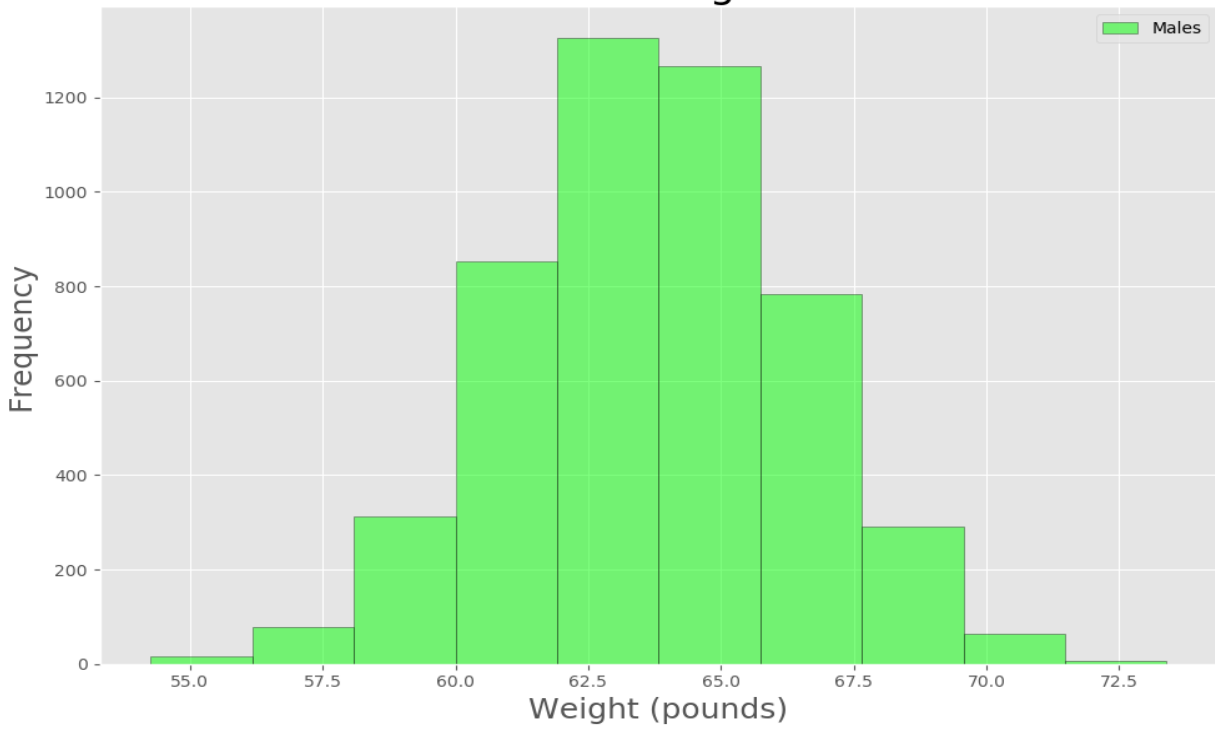


Both variables Height and Weight present a normal distribution. It can also be interesting as part of our exploratory analysis to plot the distribution of males and females in separated histograms.

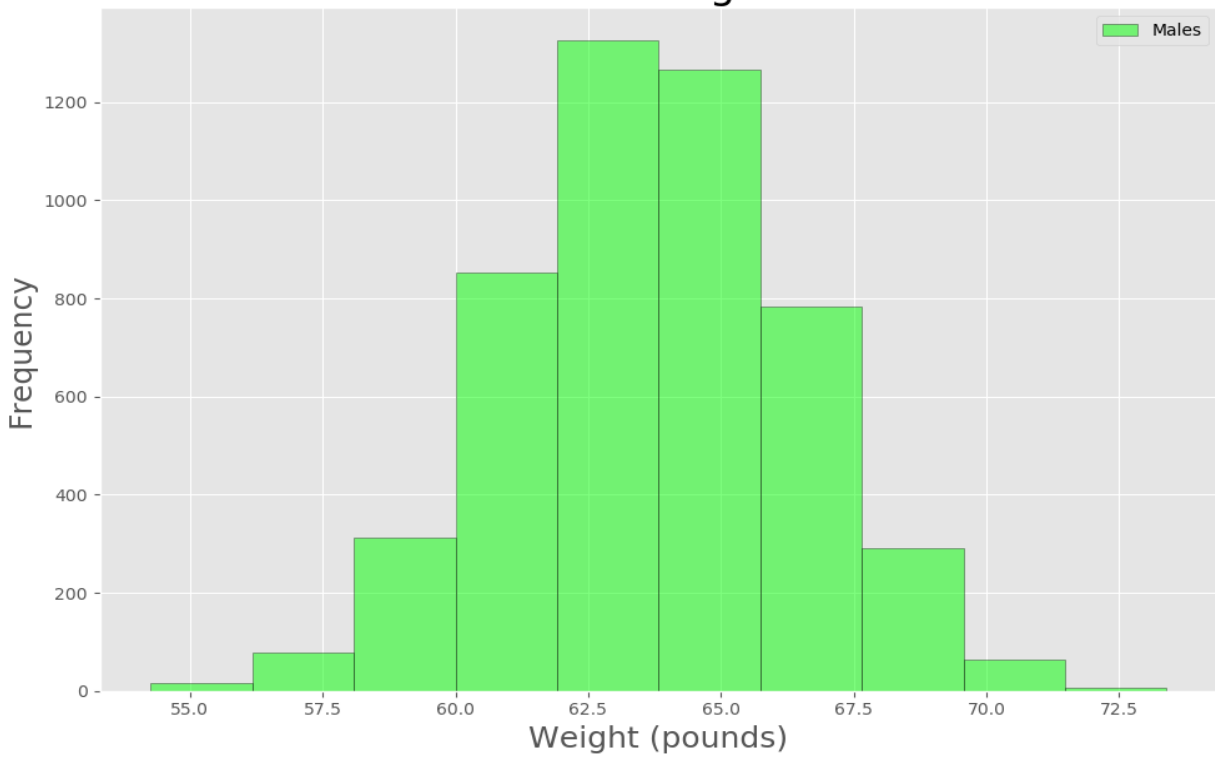


And for weight,

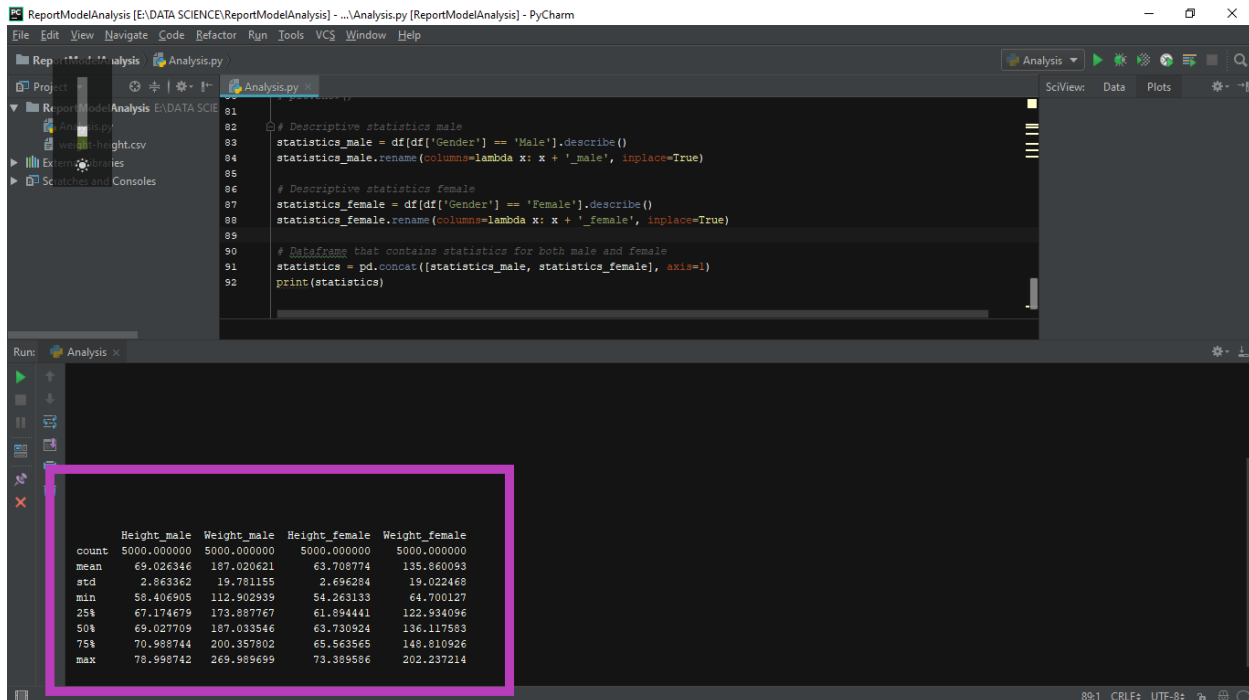
Distribution of Weight In Female



Distribution of Weight In Female



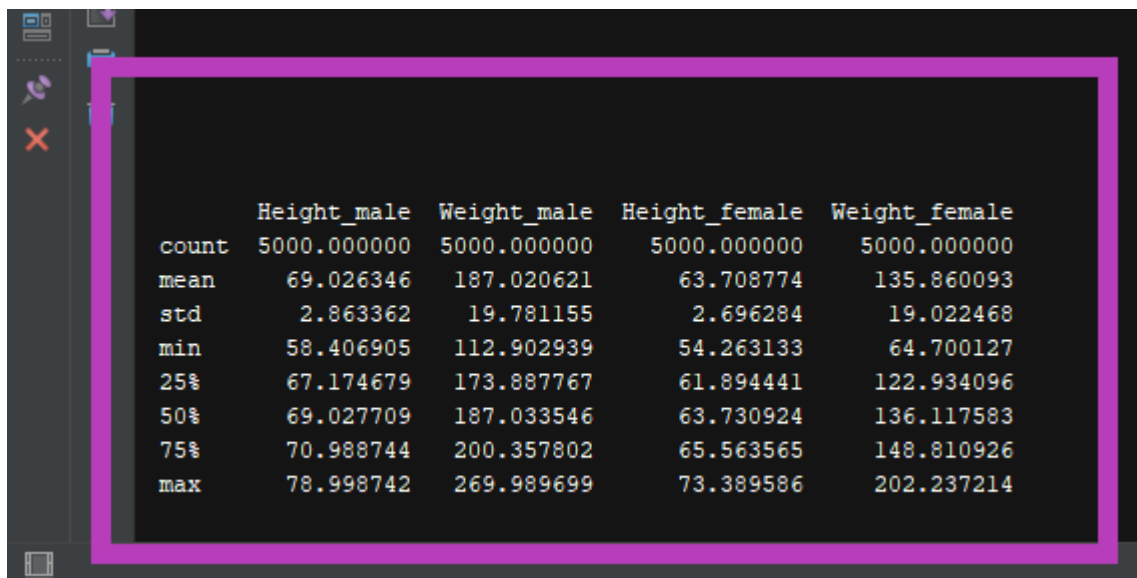
Both height and weight present a normal distribution for males and females. Although the average of each distribution is larger for males, the spread of the distributions is similar for both genders. Pandas provides a method called describe that generates descriptive statistics of a dataset (central tendency, dispersion and shape).



The screenshot shows the PyCharm IDE with a Python script named `Analysis.py` open. The script calculates descriptive statistics for 'Height' and 'Weight' for both 'Male' and 'Female' groups. The output is displayed in the Run console, showing a table of statistics for each group.

```
81 # Descriptive statistics male
82 statistics_male = df[df['Gender'] == 'Male'].describe()
83 statistics_male.rename(columns=lambda x: x + '_male', inplace=True)
84
85 # Descriptive statistics female
86 statistics_female = df[df['Gender'] == 'Female'].describe()
87 statistics_female.rename(columns=lambda x: x + '_female', inplace=True)
88
89 # DataFrame that contains statistics for both male and female
90 statistics = pd.concat([statistics_male, statistics_female], axis=1)
91 print(statistics)
```

| | Height_male | Weight_male | Height_female | Weight_female |
|-------|-------------|-------------|---------------|---------------|
| count | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 |
| mean | 69.026346 | 187.020621 | 63.708774 | 135.860093 |
| std | 2.863362 | 19.781155 | 2.696284 | 19.022468 |
| min | 58.406905 | 112.902939 | 54.263133 | 64.700127 |
| 25% | 67.174679 | 173.887767 | 61.894441 | 122.934096 |
| 50% | 69.027709 | 187.033546 | 63.730924 | 136.117583 |
| 75% | 70.988744 | 200.357802 | 65.563565 | 148.810926 |
| max | 78.998742 | 269.989699 | 73.389586 | 202.237214 |

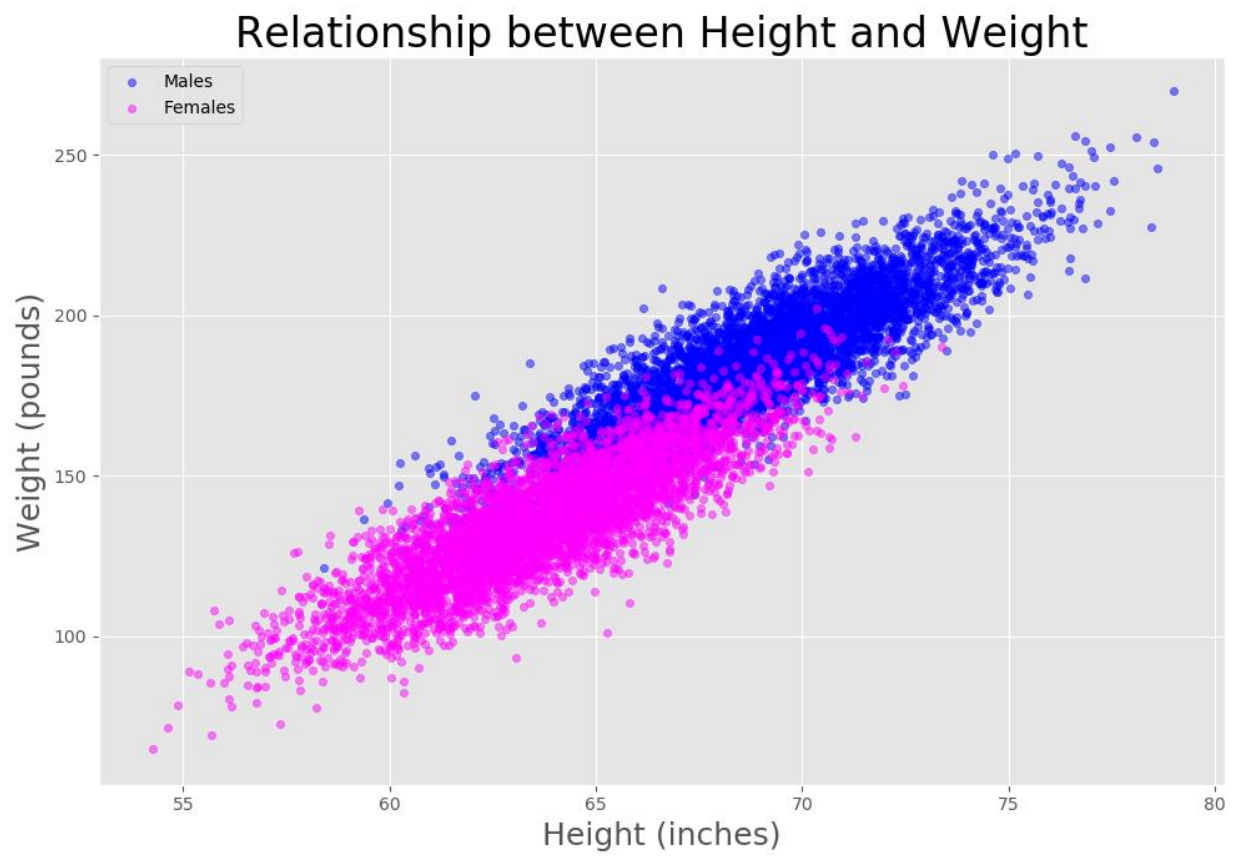
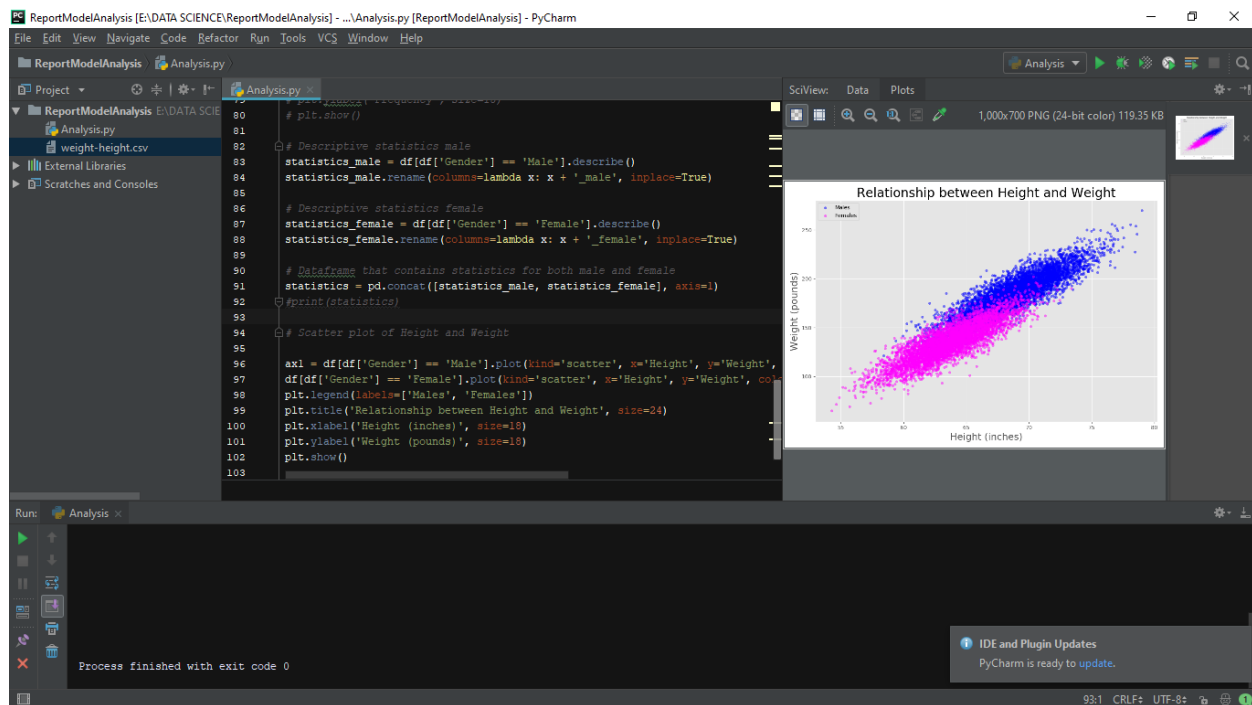


| | Height_male | Weight_male | Height_female | Weight_female |
|-------|-------------|-------------|---------------|---------------|
| count | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 |
| mean | 69.026346 | 187.020621 | 63.708774 | 135.860093 |
| std | 2.863362 | 19.781155 | 2.696284 | 19.022468 |
| min | 58.406905 | 112.902939 | 54.263133 | 64.700127 |
| 25% | 67.174679 | 173.887767 | 61.894441 | 122.934096 |
| 50% | 69.027709 | 187.033546 | 63.730924 | 136.117583 |
| 75% | 70.988744 | 200.357802 | 65.563565 | 148.810926 |
| max | 78.998742 | 269.989699 | 73.389586 | 202.237214 |

MODEL & ANALYSIS

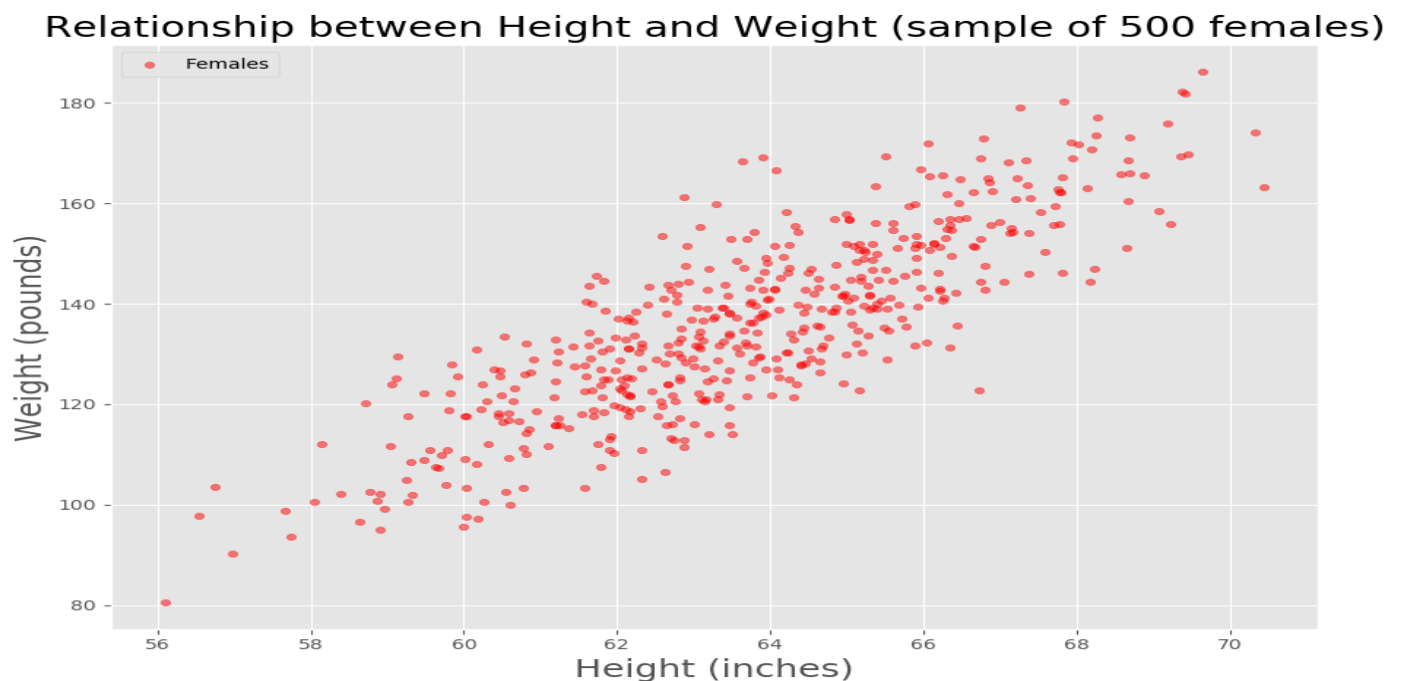
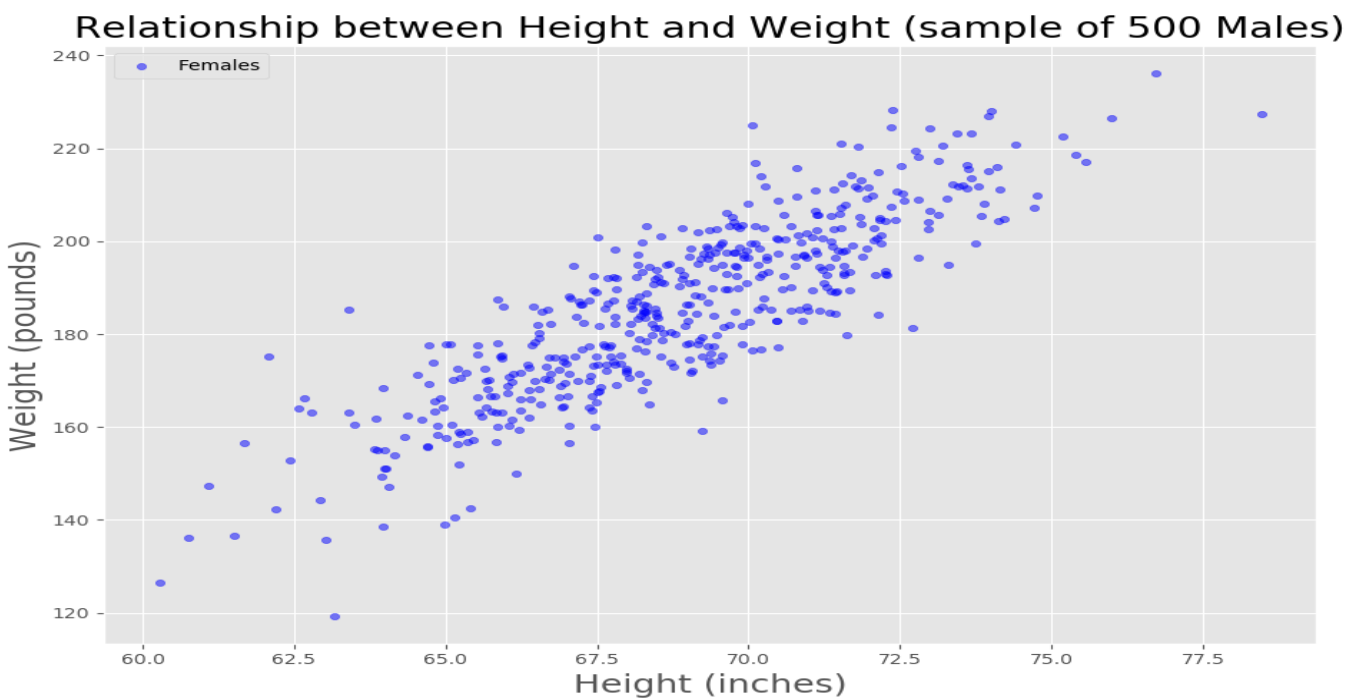


After all, now its time to decide which model we will use for prediction of weight by using height. As for now, we have two variables, one output and one input. But what if we want to include gender too. For a second, we focus on our situation, our problem is regression problem as it involves numbers.



Linear regression is a linear approach to model the relationship between a dependent variable (target variable) and one (simple regression) or more (multiple regression) independent variables. As we can see with increase in height, there is increase in weight as they are **directly proportional**.

We can study relationship of height and weight in male and female separately.



Simple linear regression is a linear approach to modeling the relationship between a dependent variable and an independent variable, obtaining a line that best fits the data.

$$\mathbf{y = a + b x}$$

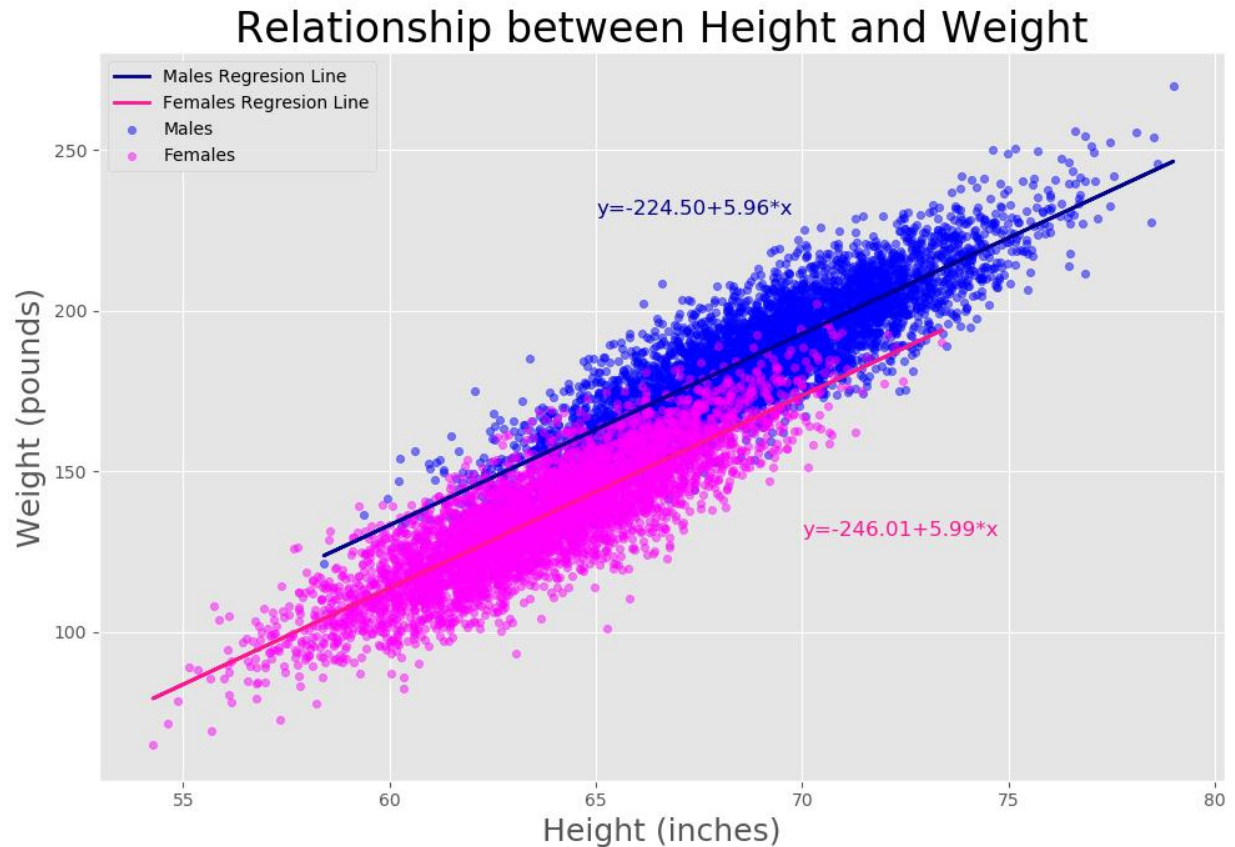
where **x** is the independent variable (height), **y** is the dependent variable (weight), **b** is the slope, and **a** is the intercept. The intercept represents the value of y when x is 0 and the slope indicates the steepness of the line. The objective is to obtain the line that best fits our data (the line that minimize the sum of square errors). The error is the difference between the real value **y** and the predicted value (**y_hat**), which is the value obtained using the calculated linear equation.

For checking error:

$$\mathbf{Error = y(real) - y(predicted)}$$

$$\mathbf{=> y(real) - y(a+bx)}$$

The numpy library function `polyfit` => `numpy.polyfit(x,y,deg)` fits a polynomial of degree *deg* to points (*x*, *y*), returning the polynomial coefficients that minimize the square error.



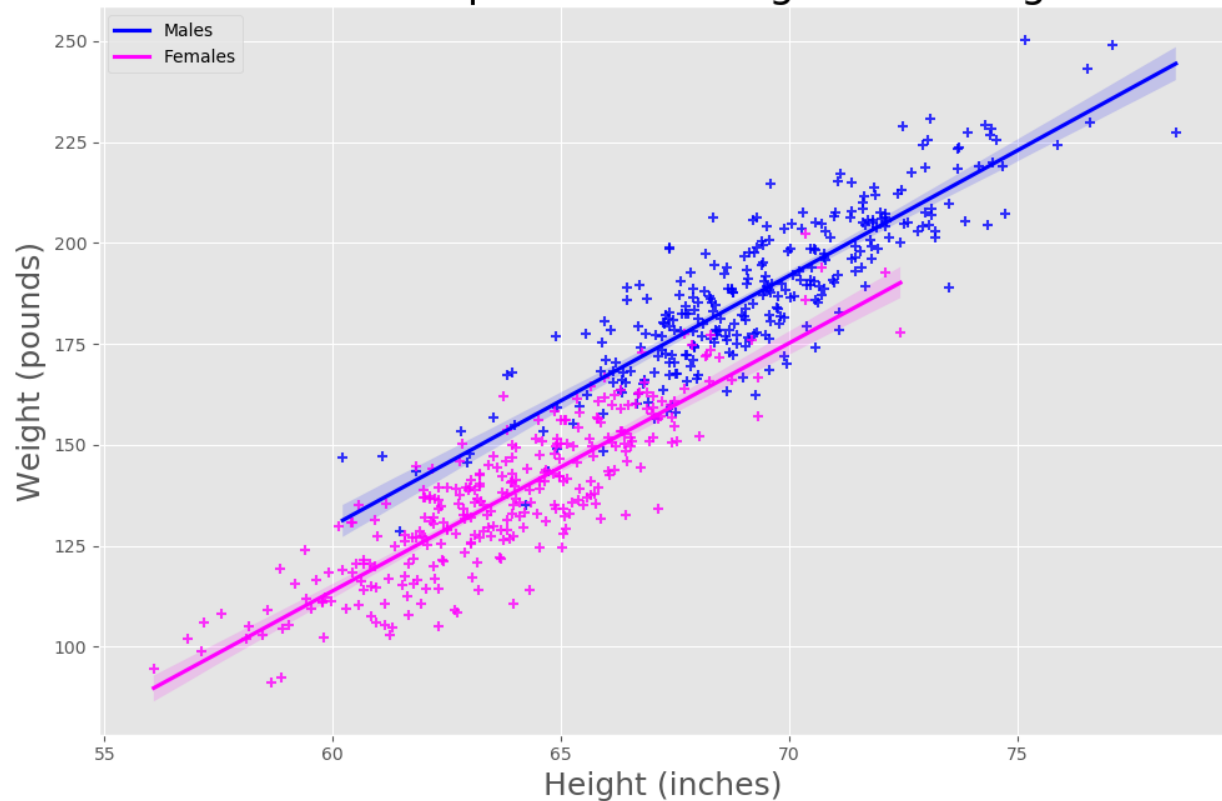
So here, we can see the clear picture of equations for both male and female.

$$\text{Male} \Rightarrow y = -224.50 + (5.96 * x)$$

$$\text{Female} \Rightarrow y = -246.01 + (5.99 * x)$$

Furthermore, let study the relationship between height and weight. Are they linear too irrespective of gender?

Relationship between Height and Weight



```
ReportModelAnalysis [E:\DATA SCIENCE\ReportModelAnalysis] - ...Analysis.py [ReportModelAnalysis] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

ReportModelAnalysis Analysis.py
Project Analysis.py
ReportModelAnalysis E:\DATA SCIENCE\ReportModelAnalysis
Analysis.py
weight-height.csv
External Libraries
Scratches and Consoles

192 # -224.49884070545772
193
194 print("Male coefficient value = ",lr_males.coef_)
195 # 5.96177381
196
197 print("\n\n")
198 df_females = df[df['Gender'] == 'Female']
199
200 # create linear regression object
201 lr_females = LinearRegression()
202
203 # fit linear regression
204 lr_females.fit(df_females[['Height']], df_females['Weight'])
205
206 # get the slope and intercept of the line best fit
207 print("Female intercept value = ",lr_females.intercept_)
208 # -246.01326574667277
209
210 print("Female coefficient value = ",lr_females.coef_)
211
```

```
Run: Analysis x
Male intercept value = -224.49884070545863
Male coefficient value = [5.96177381]

Female intercept value = -246.01326574667254
Female coefficient value = [5.99404661]

Process finished with exit code 0

Looks like you're using NumPy
Would you like to turn scientific mode on?
Use scientific mode Keep current layout...
```


Now we have got the intercept and coefficient values for both male and female model. We can also check the correlation between weight and height for male and female separately.

For females:

```
      Height    Weight
Height 1.000000 0.849609
Weight 0.849609 1.000000

Process finished with exit code 0
```

For males:

```
      Height    Weight
Height 1.000000 0.862979
Weight 0.862979 1.000000

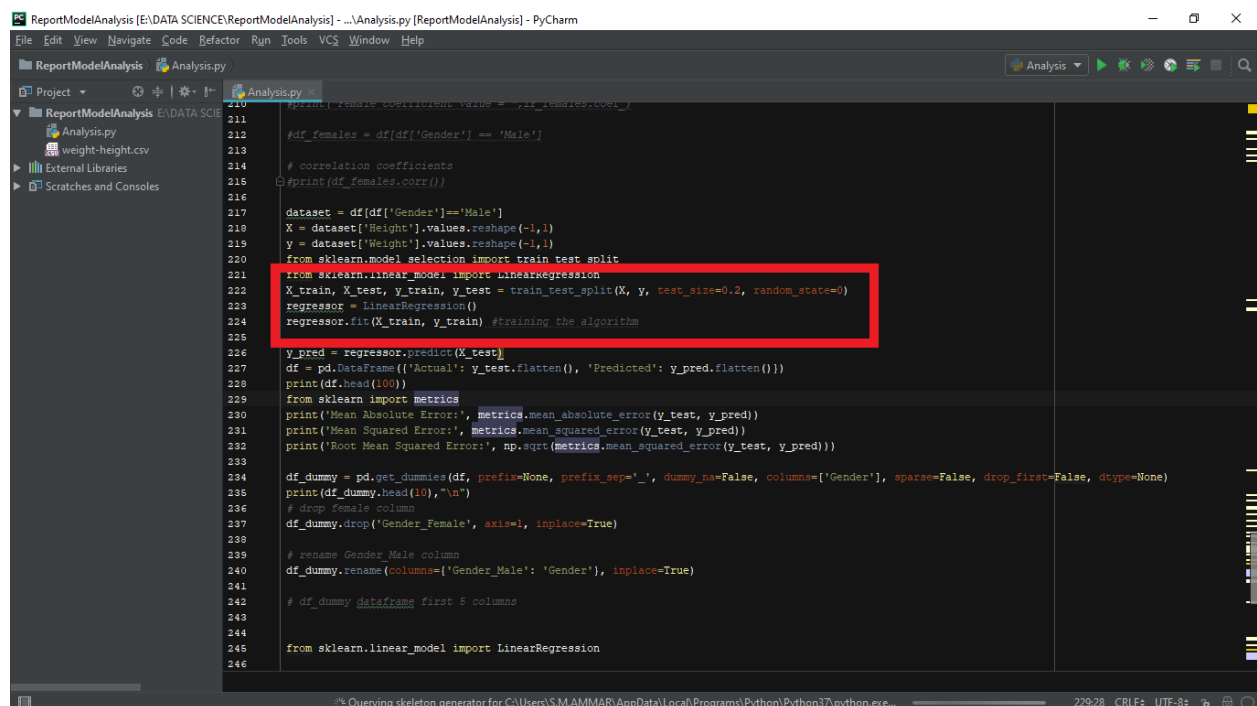
Process finished with exit code 0
```

- Females correlation coefficient: **0.849608**
- Males correlation coefficient: **0.8629788**

So, our final models for calculating weight with respect to height are:

- Males → Weight = $-224.50 + 5.96 * \text{Height}$
- Females → Weight = $-246.01 + 5.99 * \text{Height}$

Splitting Data



```
210 #print(female correlation value = r12_femalescorr)
211
212 #df_females = df[df['Gender'] == 'Male']
213
214 # correlation coefficients
215 #print(df_females.corr())
216
217 dataset = df[df['Gender']=='Male']
218 X = dataset['Height'].values.reshape(-1,1)
219 y = dataset['Weight'].values.reshape(-1,1)
220 from sklearn.model_selection import train_test_split
221 from sklearn.linear_model import LinearRegression
222 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
223 regressor = LinearRegression()
224 regressor.fit(X_train, y_train) #training the algorithm
225
226 y_pred = regressor.predict(X_test)
227 df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
228 print(df.head(100))
229 from sklearn import metrics
230 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
231 print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
232 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
233
234 df_dummy = pd.get_dummies(df, prefix=None, prefix_sep='_', dummy_na=False, columns=['Gender'], sparse=False, drop_first=False, dtype=None)
235 print(df_dummy.head(10), "\n")
236 # drop female column
237 df_dummy.drop('Gender_Female', axis=1, inplace=True)
238
239 # rename Gender_Male column
240 df_dummy.rename(columns={'Gender_Male': 'Gender'}, inplace=True)
241
242 # df_dummy dataframe first 5 columns
243
244
245 from sklearn.linear_model import LinearRegression
246
```

We can perform prediction and our accuracy will also be checked in a same way. We can also check the accuracy of our model manually too.

Accuracy of Female Model:

```

      Actual    Predicted
0   131.130447   130.899225
1   124.467491   134.662850
2   158.212861   142.937964
3   134.092031   133.653062
4   152.939283   134.474242
..      ...      ...
95  139.114234   125.815964
96  131.970135   131.749151
97  133.066749   137.123620
98  140.328416   136.289544
99  142.433424   138.590688

[100 rows x 2 columns]
Mean Absolute Error: 7.883860820364109
Mean Squared Error: 98.24910085378063
Root Mean Squared Error: 9.912068444768762

Process finished with exit code 0

```

Accuracy of male model:

```

      Actual    Predicted
0   192.470770   176.574467
1   142.252546   146.465472
2   213.457233   199.793112
3   152.896965   180.213770
4   163.495203   178.241772
..      ...      ...
95  182.856213   195.514418
96  201.795504   186.069621
97  176.936187   185.631572
98  184.833520   193.132403
99  166.171185   173.936738

[100 rows x 2 columns]
Mean Absolute Error: 7.946830514126609
Mean Squared Error: 99.2717818984359
Root Mean Squared Error: 9.963522564757703

Process finished with exit code 0

```

So, you can see these are not perfect models but they are good enough to predict weight in real life (close to original ones).

Now we can also create a model with gender and height as input.



| | Height | Weight | Gender_Female | Gender_Male |
|---|-----------|------------|---------------|-------------|
| 0 | 73.847017 | 241.893563 | 0 | 1 |
| 1 | 68.781904 | 162.310473 | 0 | 1 |
| 2 | 74.110105 | 212.740856 | 0 | 1 |
| 3 | 71.730978 | 220.042470 | 0 | 1 |
| 4 | 69.881796 | 206.349801 | 0 | 1 |
| 5 | 67.253016 | 152.212156 | 0 | 1 |
| 6 | 68.785081 | 183.927889 | 0 | 1 |
| 7 | 68.348516 | 167.971110 | 0 | 1 |
| 8 | 67.018950 | 175.929440 | 0 | 1 |
| 9 | 63.456494 | 156.399676 | 0 | 1 |

Intercept Value = -244.92350252069937

Coefficient values = [5.97694123 19.37771052]

Process finished with exit code 0

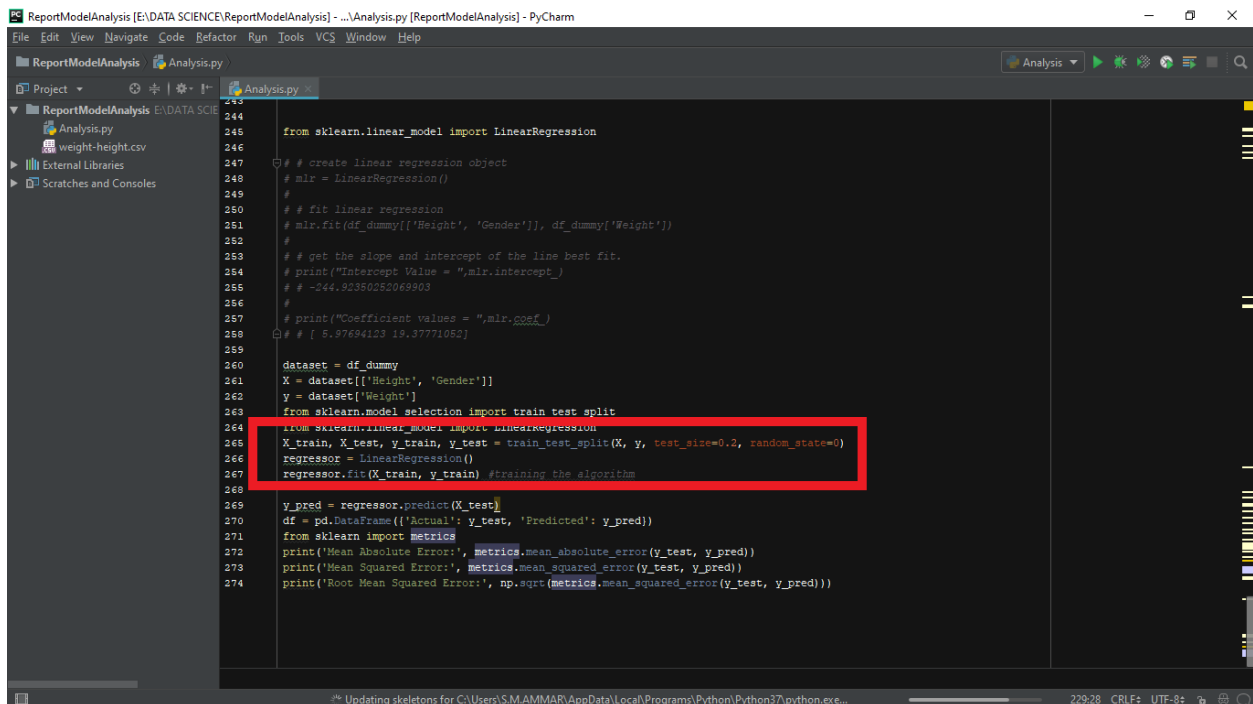
New Model:

$$\text{Weight} = -244.9235 + 5.9769 * \text{Height} + 19.3777 * \text{Gender}$$

$$\text{Male} \rightarrow \text{Weight} = -244.9235 + 5.9769 * \text{Height} + 19.3777 * 1 = -225.5458 + 5.9769 * \text{Height}$$

$$\text{Female} \rightarrow \text{Weight} = -244.9235 + 5.9769 * \text{Height} + 19.3777 * 0 = -244.9235 + 5.9769 * \text{Height}$$

Data splitting and testing:



```
ReportModelAnalysis [E:\DATA SCIENCE\ReportModelAnalysis] - ...Analysis.py [ReportModelAnalysis] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

ReportModelAnalysis
  Analysis.py
  weight-height.csv
  External Libraries
  Scratches and Consoles

243
244
245 from sklearn.linear_model import LinearRegression
246
247 # create linear regression object
248 # mlr = LinearRegression()
249 #
250 # fit linear regression
251 # mlr.fit(df_dummy[['Height', 'Gender']], df_dummy['Weight'])
252 #
253 # get the slope and intercept of the line best fit.
254 # print("Intercept Value = ",mlr.intercept_)
255 # # -244.92350252069903
256 #
257 # print("Coefficient values = ",mlr.coef_)
258 # # [ 5.97694123 19.37771052]
259
260 dataset = df_dummy
261 X = dataset[['Height', 'Gender']]
262 y = dataset['Weight']
263 from sklearn.model_selection import train_test_split
264
265 # train the model
266 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
267 regressor = LinearRegression()
268 regressor.fit(X_train, y_train) #training the algorithm
269
270 y_pred = regressor.predict(X_test)
271 df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
272 from sklearn import metrics
273 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
274 print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
275 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

The image shows a PyCharm IDE window titled 'ReportModelAnalysis [E:\DATA SCIENCE\ReportModelAnalysis] - Analysis.py [ReportModelAnalysis] - PyCharm'. The editor displays a Python script for linear regression analysis. The script imports necessary libraries, loads a dataset, splits it into training and testing sets, fits a LinearRegression model, and calculates performance metrics. The output window shows the resulting data and metrics.

```
259 dataset = df_dummy
260 X = dataset[['Height', 'Gender']]
261 y = dataset['Weight']
262
263 from sklearn.model_selection import train_test_split
264 from sklearn.linear_model import LinearRegression
265 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
266 regressor = LinearRegression()
267 regressor.fit(X_train, y_train) #training the algorithm
268
269 y_pred = regressor.predict(X_test)
270 df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
271 from sklearn import metrics
272 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

Run: Analysis x

| | Height | Weight | Gender_Female | Gender_Male |
|---|-----------|------------|---------------|-------------|
| 0 | 73.847017 | 241.889563 | 0 | 1 |
| 1 | 68.781904 | 162.310473 | 0 | 1 |
| 2 | 74.110105 | 212.740856 | 0 | 1 |
| 3 | 71.730978 | 220.042470 | 0 | 1 |
| 4 | 65.881796 | 206.349801 | 0 | 1 |
| 5 | 67.253016 | 152.212156 | 0 | 1 |
| 6 | 68.785081 | 183.927889 | 0 | 1 |
| 7 | 68.348516 | 167.971110 | 0 | 1 |
| 8 | 67.018950 | 175.929440 | 0 | 1 |
| 9 | 63.456494 | 156.399676 | 0 | 1 |

Mean Absolute Error: 7.955916525326745
Mean Squared Error: 97.87152220196162
Root Mean Squared Error: 9.893003699684016

Process finished with exit code 0

The End