МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторные работы по дисциплине

«Введение в машинное обучение»

Выполнил:

Студент гр.09-305

Шарафеев Марат

Преподаватель:

Шустова Е.П.

Казань-2024

# **Задание №8. Метод деревьев решений.**

**Выполните следующее задачи:**

1. Подберите свои данные для решения задачи классификации с помощью дерева решений. Для этого надо воспользоваться известными репозиториями данных (см., например, http://archive.ics.uci.edu/ml/index.php). К указанному ниже сроку закрепите эти данные в Teams через ответить в начатой мною беседе. Закрепите это означает, что надо привести краткое описание выбранных Вами данных и интернет-ссылку на эти данные. Перед поиском своих данных посмотрите в указанной выше беседе какие данные уже заняты другими студентами.
2. Установите необходимые внешние библиотеки.
3. Приведите скриншоты путей, в которые Вы установили Python, PyCharm и внешние библиотеки.
4. Импортируйте необходимые пакеты и классы:
5. Осуществите описание Вашего набора данных. Загрузите Ваши данные и распечатайте первые 10 наблюдений.
6. Подготовьте данные для работы. А именно: определите, есть ли в наборе данных пропущенные значения; если в данных присутствуют пропущенные значения, то либо удалите соответствующие наблюдения, либо рассчитайте их значение известным Вам методом; удалите выбросы, если они есть.
7. Рассчитайте и выведите матрицу корреляции для Ваших данных. Укажите какой признак Вы берете за переменную отклика, а какие за факторы. Обоснуйте свое решение.
8. Сформируйте обучающую и тестовую выборки с помощью метода train\_test\_split из scikitlearn (например так: train\_test\_split(X, y, test\_size = 0.2, random\_state=9).
9. Если число факторов m <= 2, то визуализируйте данные (постройте облако точек).
10. Отделите от Ваших данных прогнозное значение (переменную отклика), то есть удалите из Ваших данных столбец с прогнозными значениями.
11. Постройте натренированную модель дерева решений методом [DecisionTreeClassifier](https://scikit--learn-org.translate.goog/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?_x_tr_sl=en&_x_tr_tl=ru&_x_tr_hl=ru&_x_tr_pto=ajax,sc,elem%23sklearn.tree.DecisionTreeClassifier) . Сделайте выводы из полученного результата.
12. Исследуйте дерево решений; если позволяет размерность, постройте его график.
13. Найдите средние абсолютную и квадратичную оценку ошибки модели и качество модели.
14. Сравните теоретические и эмпирические значения переменной отклика на каких-нибудь двух значениях наблюдений из тестовой выборки. Сделайте вывод.
15. Приведите полный код для решения задачи. Код сопровождайте комментариями.
16. Оформите отчёт о результатах исследования и прикрепите его в ЭОР «Введение в машинное обучение на Python» по ссылке <https://repository.kpfu.ru/?p_id=236426> в указанный ниже период до или во время своей пары.
17. К этому же заданию прикрепите файл с данными и py-файл с программой.

**Ход работы:**

1. Для задачи классификации были выбраны данные о предсказании возраста морского ушка (моллюск) по физическим измерениям. Автор набора данных утверждает, что процесс нахождения возраста моллюска весьма трудоемкий и по этой причине был выпущен датасет, который был призван для разработки модели, которая бы по внешним физическим показателем мог бы с достаточно большой точностью определить возраст ушка. Набор данных содержит 4177 экземпляров. С оригинал датасета можно получить по [ссылке](http://archive.ics.uci.edu/ml/datasets/Abalone).

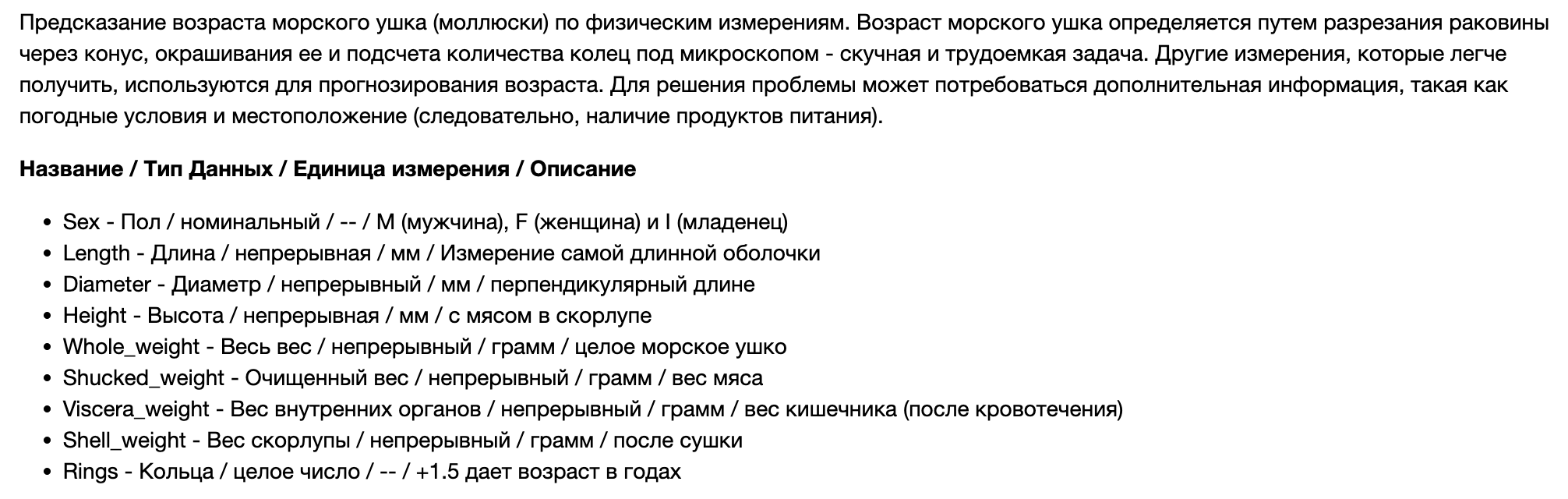
Данные содержат следующие характеристики:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Тип данных** | **Ед. изм.** | **Описание** |
| Sex - Пол | номинативная | – | M (мужчина), F (женщина) и I (младенец) |
| Length - Длина | непрерывная | мм | Измерение самой длинной оболочки |
| Diameter - Диаметр | непрерывный | мм | перпендикулярный длине |
| Height - Высота | непрерывная | мм | с мясом в скорлупе |
| Whole\_weight - Весь вес | непрерывный | грамм | целое морское ушко |
| Shucked\_weight - Очищенный вес | непрерывный | грамм | вес мяса |
| Viscera\_weight - Вес внутренних органов | непрерывный | грамм | вес кишечника (после кровотечения) |
| Shell\_weight - Вес скорлупы | непрерывный | грамм | после сушки |
| Rings - Кольца | целое число | – | +1.5 дает возраст в годах |

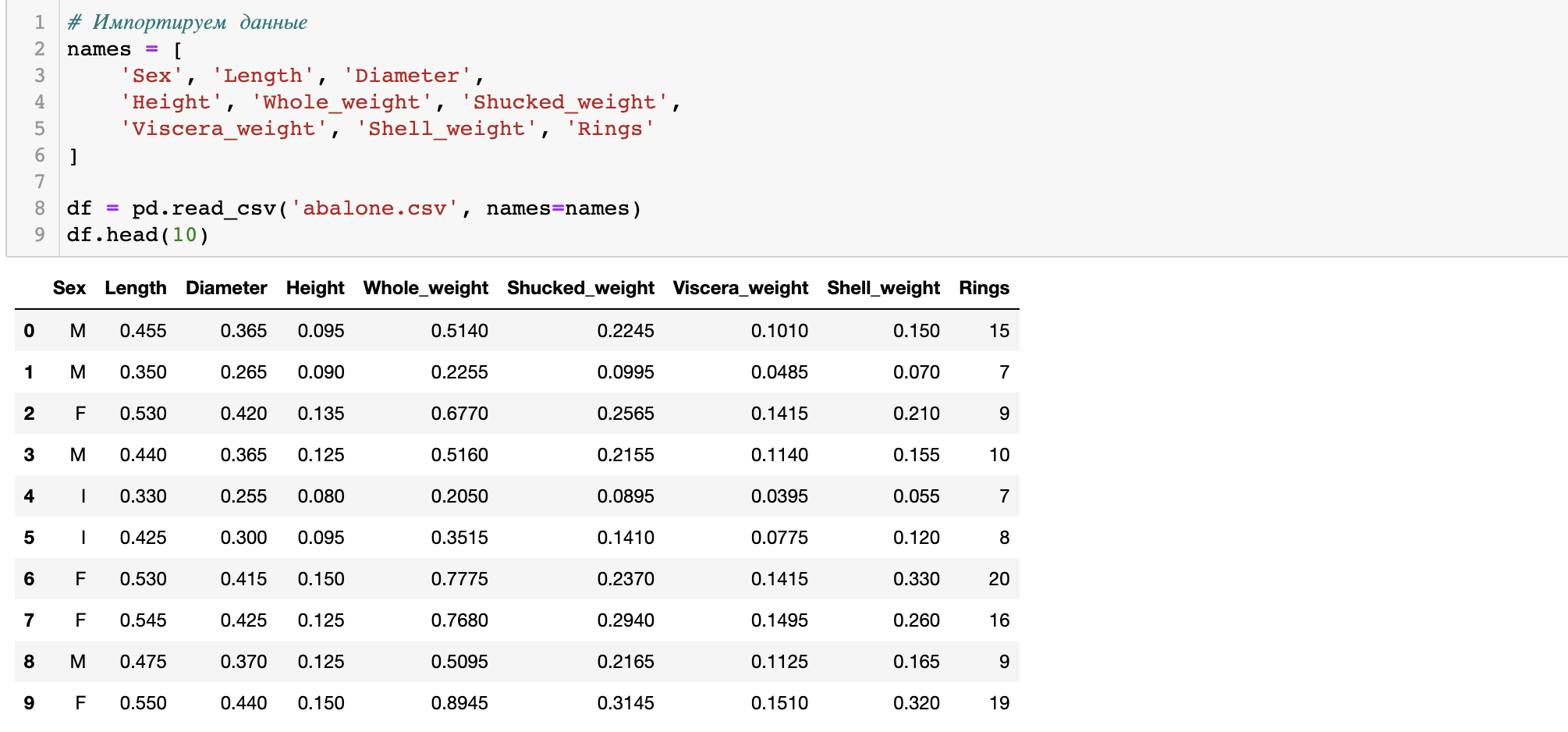
1. Импортируем все необходимые библиотеки:



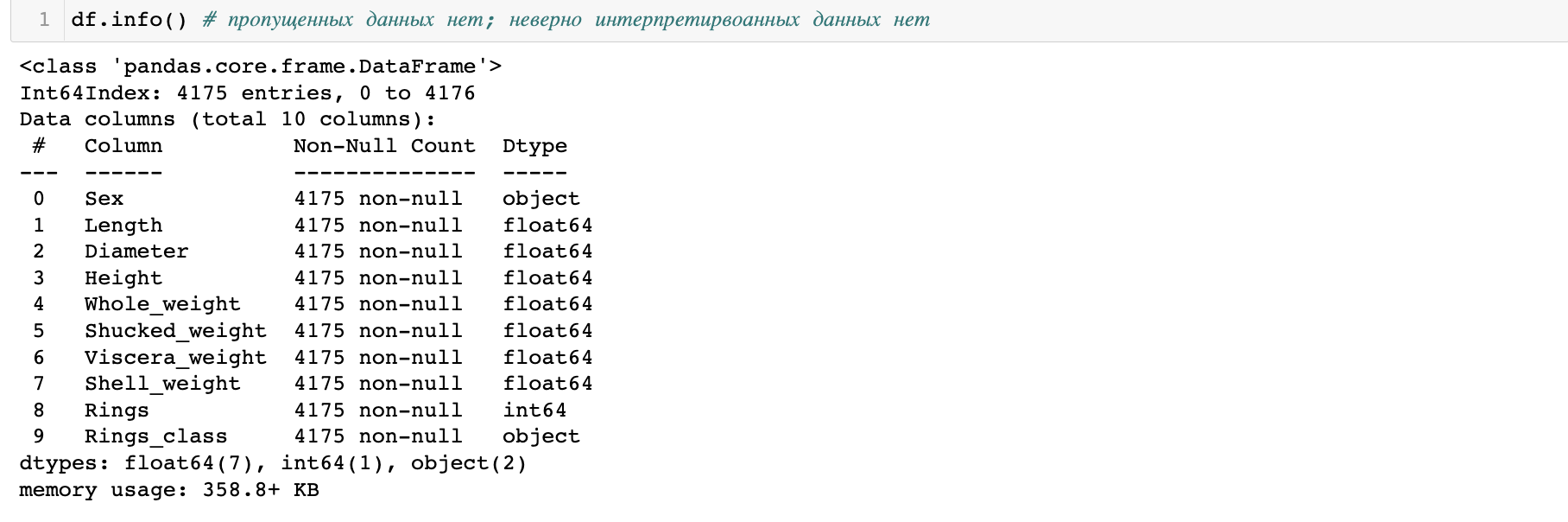
1. Осуществим описание набора данных.



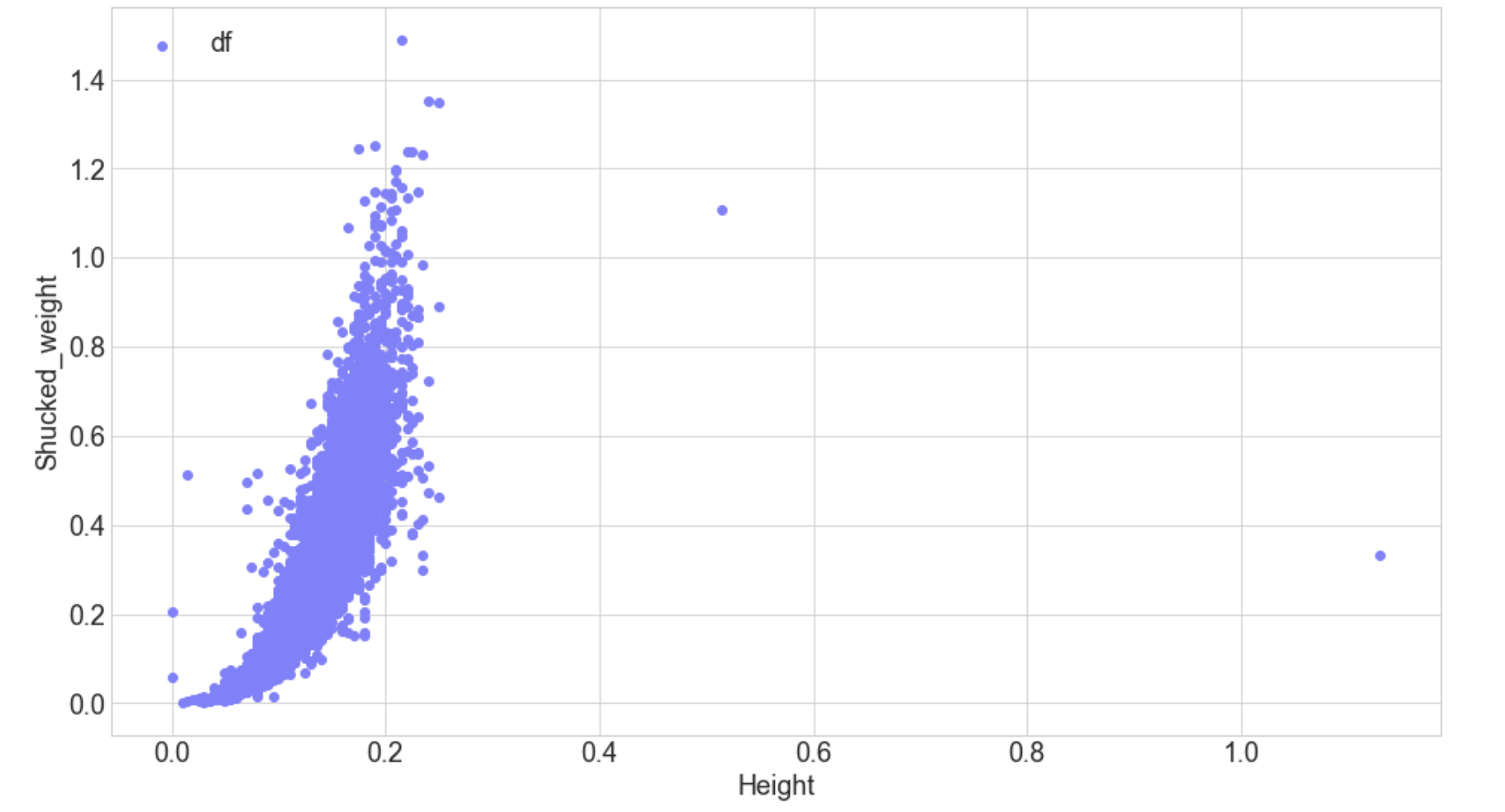
Загрузим наши данные и распечатаем первые 10 наблюдений.



1. Подготовим данные для работы. Определим имеются ли в наших данных пропуски:



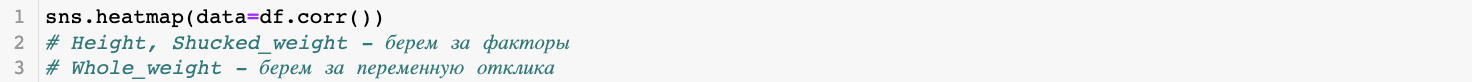
Видно, что пропусков нет и все данные корректно интерпретированы в свои типы. Теперь проверим визуально на выбросы.

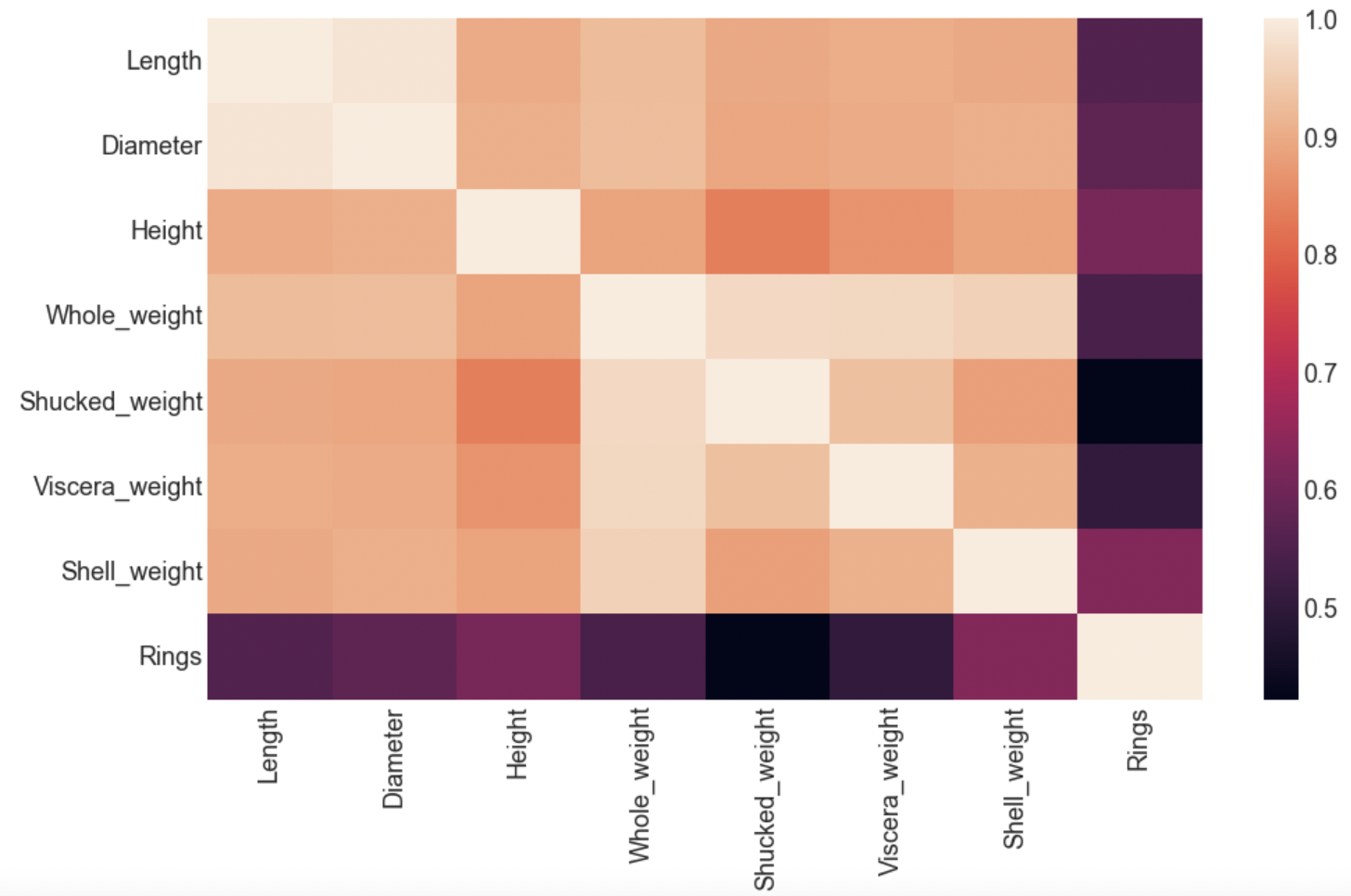
 

Видно, что имеется два выброса. Удалим их.

Рисунок 64

1. Рассчитаем матрицу корреляции.





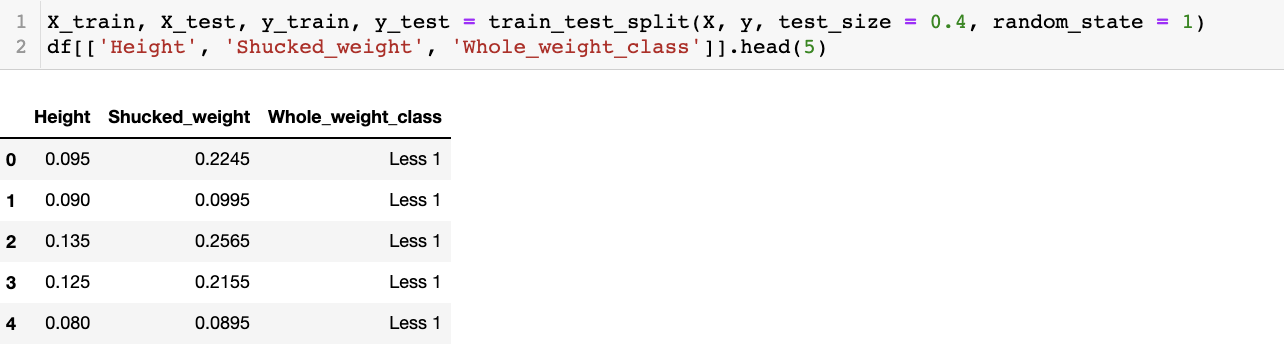
Переменную отклика Whole\_weight разделим на две группы «Over 1» и «Less 1»

Рисунок 12

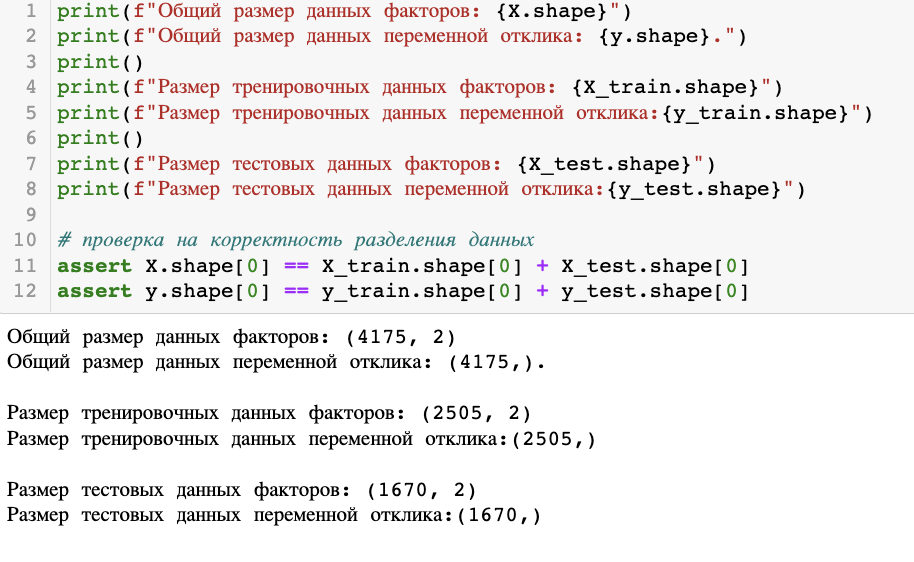
Переменные Height (Высота), Shucked\_weight (Очищенный вес) - менее всего коррелируют, по этой причине мы берем их за факторы, а Whole\_weight\_class - берем за переменную отклика

Рисунок 68

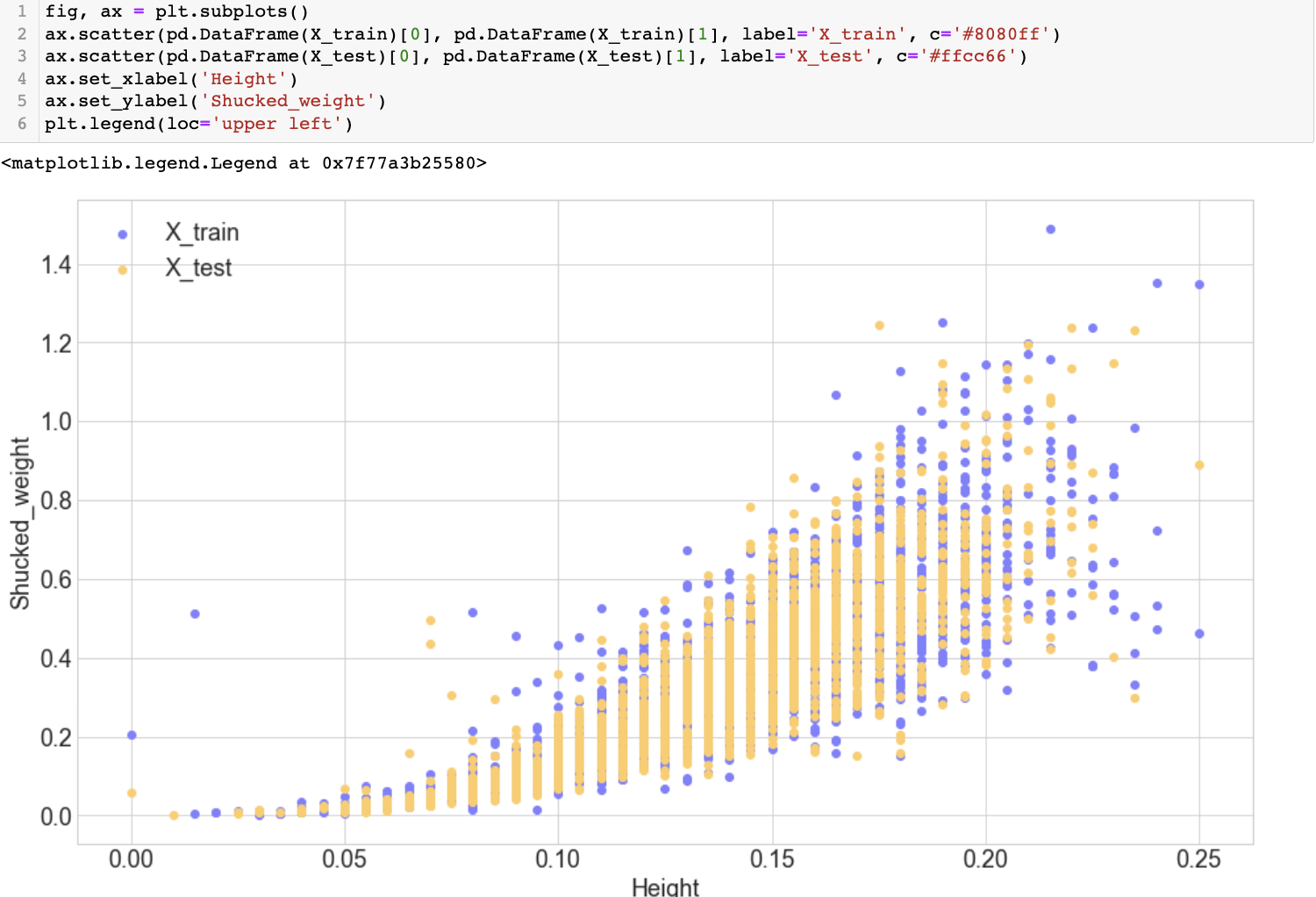
1. Сформируем обучающую и тестовую выборки.



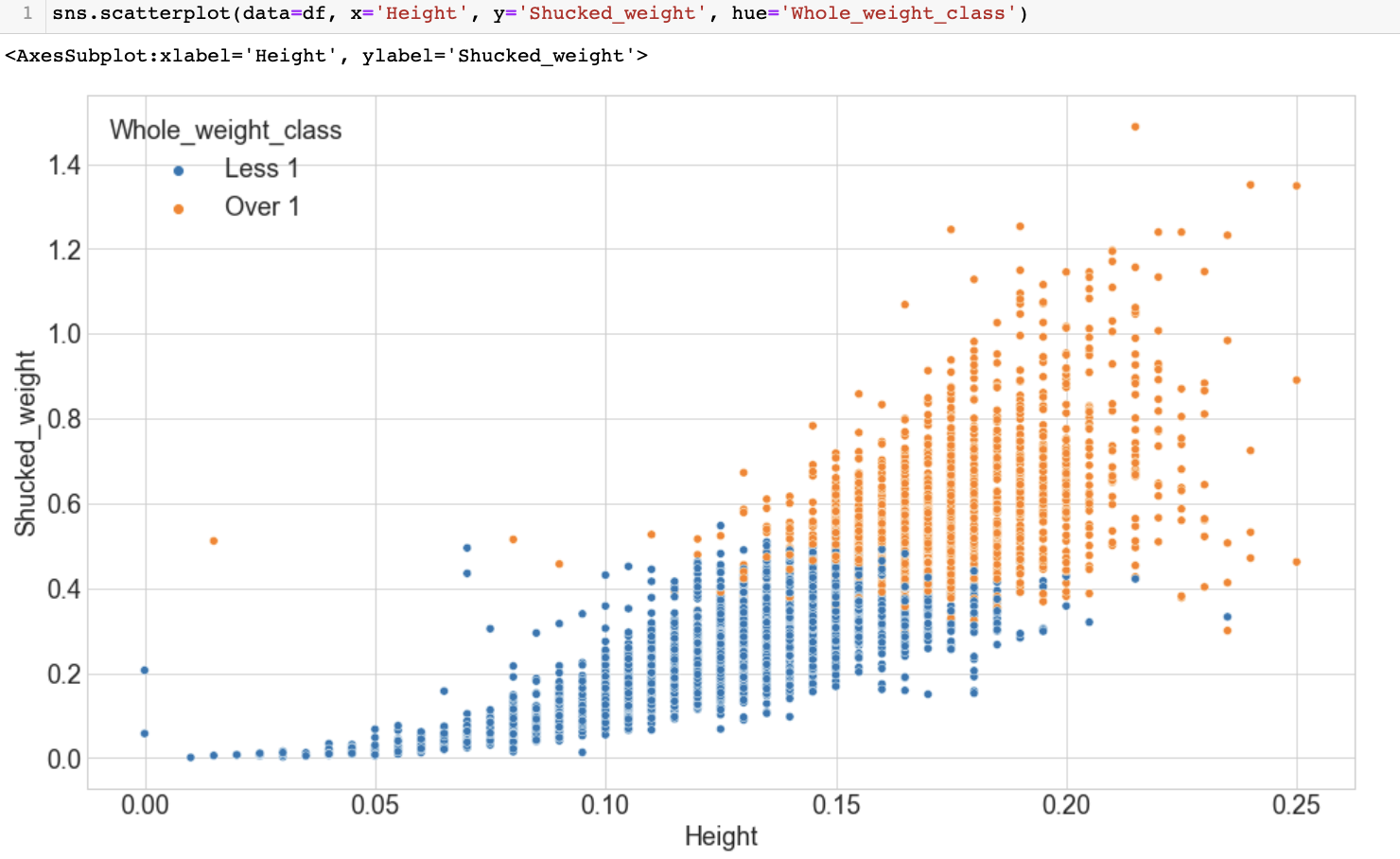
Здесь мы разделили наши данные на обучающую и тестовую и сразу же разделили факторы и отклики.



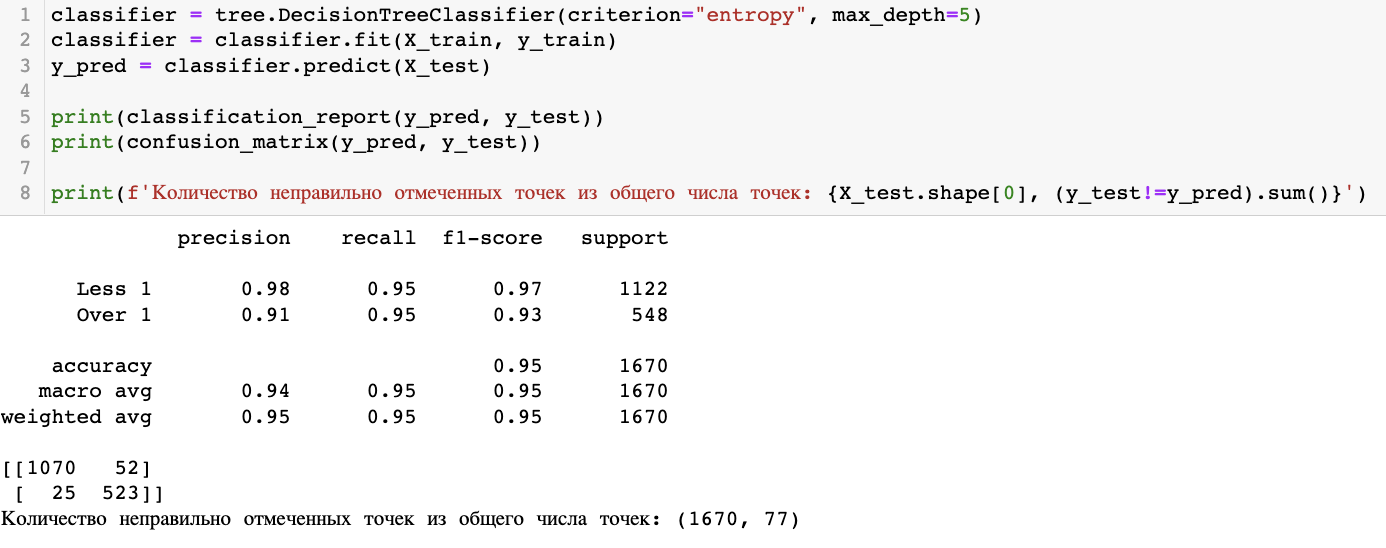
1. Визуализируем наши данные:



На графике разным цветом отмечены тестовые и тренировочные данные. Теперь посмотрим как распределены разные классы.



1. Построим натренированную модель методом решающих деревьев.

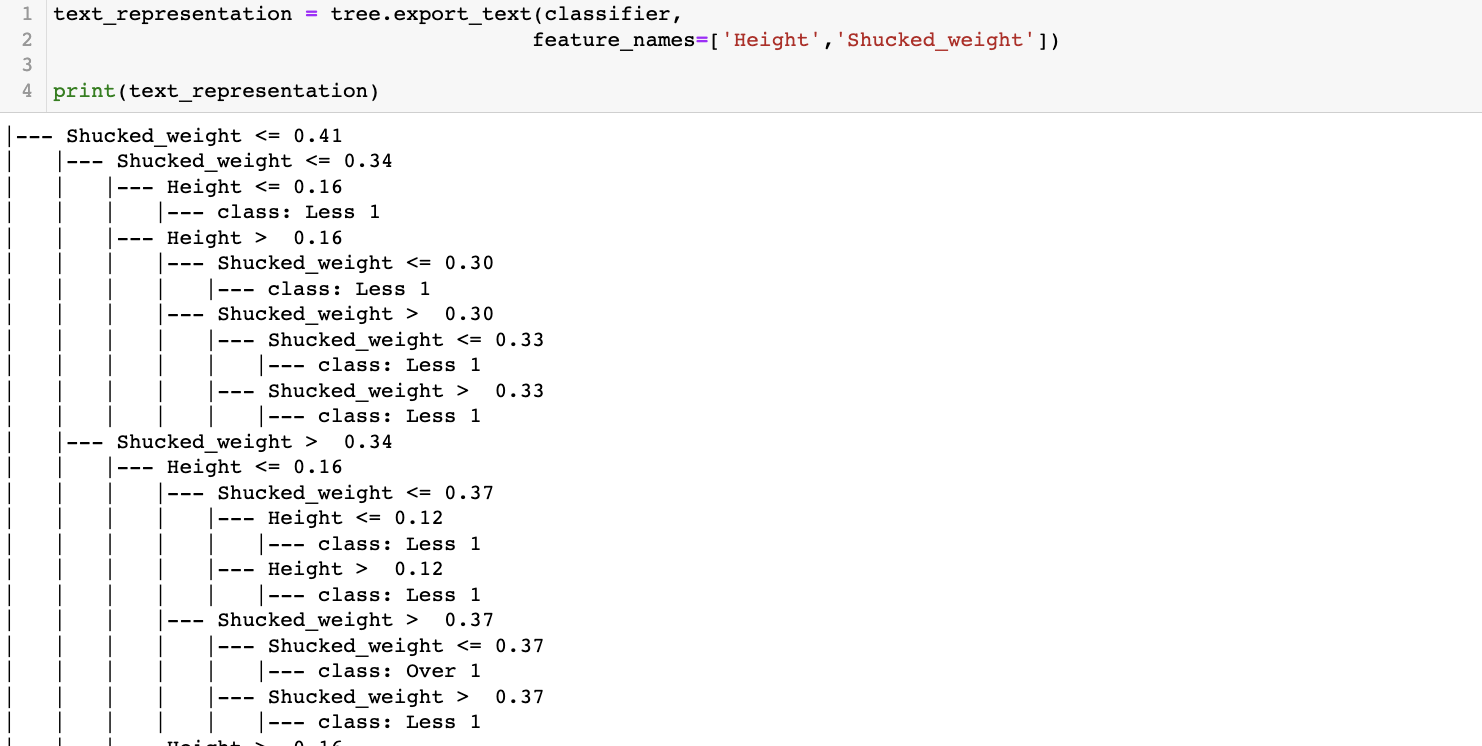


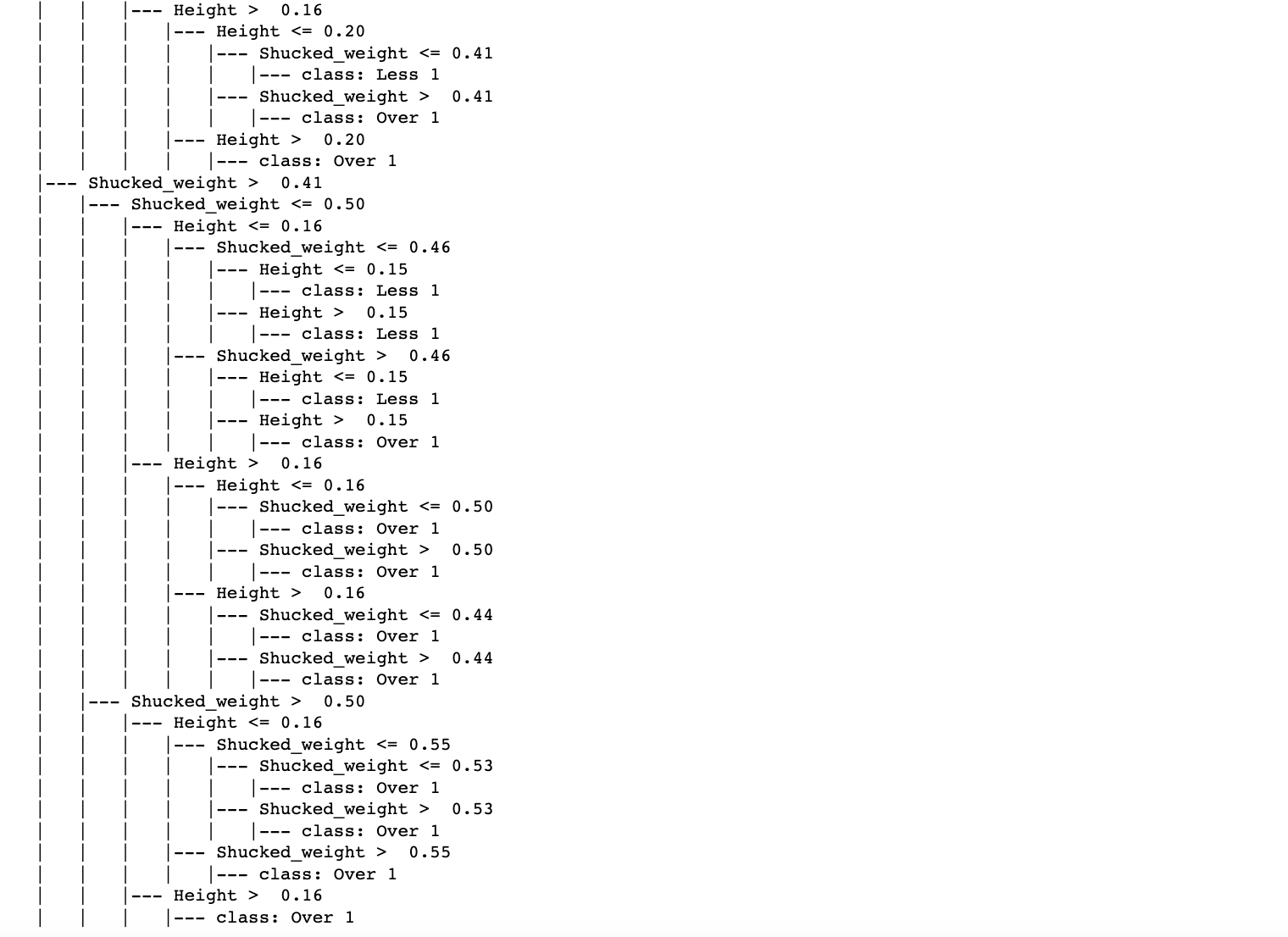
Мы выставили способ построения дерева на основе показателя энтропии, а также ограничили глубину дерева до 5 дабы не переобучить модель. Глубину дерева подбирали исходя из минимизации количества допущенных ошибок в натренированной модели.

1. Продемонстрируем полученное дерево



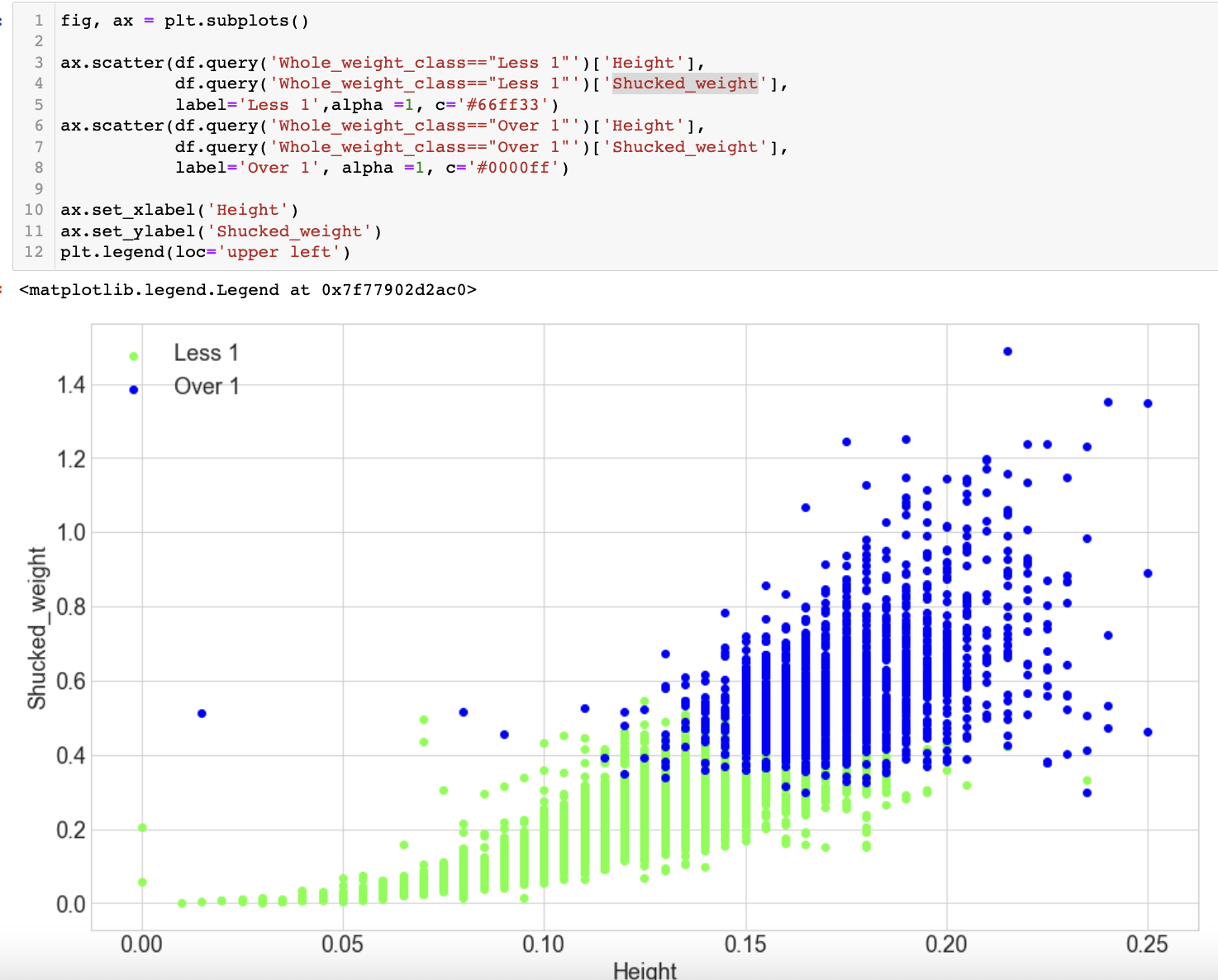
Рисунок можно посмотреть и прочитать при желании отдельно, однако в рамках лабораторной работы намного более читабельным будет текстовый формат представления дерева.



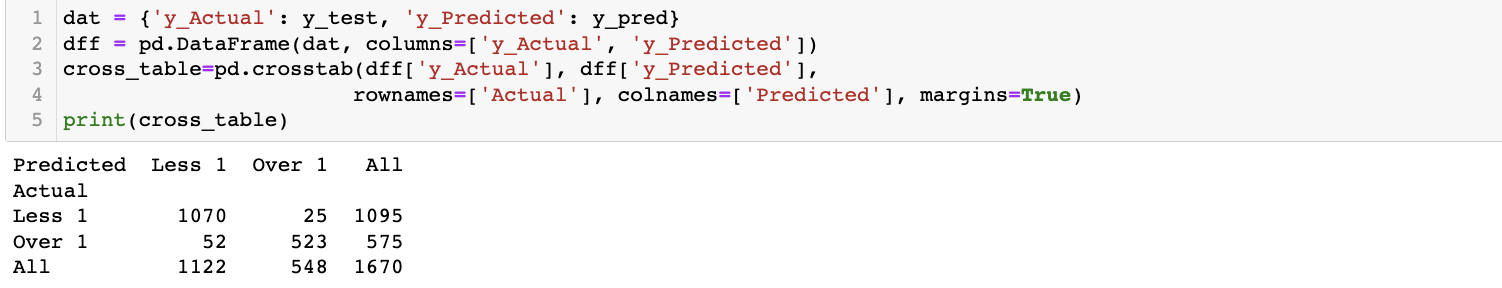


По графику и следуя дереву видим, что сначала мы делим наши данные по Shucked\_weight<= 0.41 – большинство легких моллюсков (тяжелые присутствуют), и Shucked\_weight > 0.41 - тяжелые моллюски (легкие присутствуют). Далее по каждой из ветвей Shucked\_weight<= 0.41: Shucked\_weight <= 0.34 - большинство легких (тяжелые присутствуют), Shucked\_weight > 0.41 - большинство тяжелых (легкие присутствуют); Shucked\_weight > 0.5: Height <= 0.16 - большинство легкие (тяжелые присутствуют), Height > 0.16 - большинство тяжелые (легкие присутствуют).

Если дальше следовать этой логике придем к ситуации, когда в каждом сегменте останутся только один класс. Однако мы специально ограничили глубину дерева дабы не переучить модель.



1. Для оценки точности используем кросс-валидационную таблицу:



Наша выборка состояла из 1670 объектов – 548 объекта первого класса, 1122 объектов второго класса. Модель допустила ошибки в 77 случаях из 1670, что соответствует точности прогноза в 95%. Таким образом, мы допускаем, что наличие информации о очищенном весе и высоте морских ушков дает достаточно хорошую оценку о том, весе моллюска.

1. Вывод: Таким образом, мы допускаем, что наличие информации о очищенном весе и высоте морских ушков дает достаточно хорошую оценку о том, легкий моллюск или старый.

**Приложение:**

|  |
| --- |
| # Импортируем необходимые библиотеки для анализа данных  # Библиотеки работы с данными  import pandas as pd  import numpy as np  # Библиотки визуализации  import matplotlib.pyplot as plt  import seaborn as sns  # Настройка графиков  plt.style.use('seaborn-whitegrid')  plt.rcParams["figure.figsize"] = (16, 9)  plt.rcParams.update({'font.size': 18})  # Библиотека машинного обучения  import sklearn  from sklearn.model\_selection import train\_test\_split  from sklearn import svm  from sklearn import tree  from sklearn.tree import DecisionTreeClassifier  from sklearn.metrics import classification\_report  from sklearn.metrics import confusion\_matrix  from sklearn.metrics import accuracy\_score |
| # Импортируем данные  names = [  'Sex', 'Length', 'Diameter',  'Height', 'Whole\_weight', 'Shucked\_weight',  'Viscera\_weight', 'Shell\_weight', 'Rings'  ]  df = pd.read\_csv('abalone.csv', names=names)  df.head(10) |
| df.info() # пропущенных данных нет; неверно интерпретирвоанных данных нет |
| fig, ax = plt.subplots()  ax.scatter(df.Height, df.Shucked\_weight, label='df', c='#8080ff')  ax.set\_xlabel('Height')  ax.set\_ylabel('Shucked\_weight')  plt.legend(loc='upper left') |
| df = df.drop(index=list(df.query("Height>0.4").index)) |
| sns.heatmap(data=df.corr(numeric\_only=True))  # Height, Shucked\_weight - менее всего коррелируют  # Height, Shucked\_weight - берем за факторы |
| df['Whole\_weight\_class'] = np.where(df.Whole\_weight > 1, 'Over 1', 'Less 1') |
| X, y = df[['Height', 'Shucked\_weight']].values, df['Whole\_weight\_class'] |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4, random\_state = 1)  df[['Height', 'Shucked\_weight', 'Whole\_weight\_class']].head(5) |
| print(f"Общий размер данных факторов: {X.shape}")  print(f"Общий размер данных переменной отклика: {y.shape}.")  print()  print(f"Размер тренировочных данных факторов: {X\_train.shape}")  print(f"Размер тренировочных данных переменной отклика:{y\_train.shape}")  print()  print(f"Размер тестовых данных факторов: {X\_test.shape}")  print(f"Размер тестовых данных переменной отклика:{y\_test.shape}")  # проверка на корректность разделения данных  assert X.shape[0] == X\_train.shape[0] + X\_test.shape[0]  assert y.shape[0] == y\_train.shape[0] + y\_test.shape[0] |
| fig, ax = plt.subplots()  ax.scatter(pd.DataFrame(X\_train)[0], pd.DataFrame(X\_train)[1], label='X\_train', c='#8080ff')  ax.scatter(pd.DataFrame(X\_test)[0], pd.DataFrame(X\_test)[1], label='X\_test', c='#ffcc66')  ax.set\_xlabel('Height')  ax.set\_ylabel('Shucked\_weight')  plt.legend(loc='upper left') |
| sns.scatterplot(data=df, x='Height', y='Shucked\_weight', hue='Whole\_weight\_class') |
| classifier = tree.DecisionTreeClassifier(criterion="entropy", max\_depth=6)  classifier = classifier.fit(X\_train, y\_train)  y\_pred = classifier.predict(X\_test)  print(classification\_report(y\_pred, y\_test))  print(confusion\_matrix(y\_pred, y\_test))  print(f'Количество неправильно отмеченных точек из общего числа точек: {X\_test.shape[0], (y\_test!=y\_pred).sum()}') |
| fig = plt.figure(figsize=(50,20))  \_ = tree.plot\_tree(classifier,  fontsize=8,  feature\_names=['Height','Shucked\_weight'],  filled=True) |
| text\_representation = tree.export\_text(classifier,  feature\_names=['Height','Shucked\_weight'])  print(text\_representation) |
| fig, ax = plt.subplots()  ax.scatter(df.query('Whole\_weight\_class=="Less 1"')['Height'],  df.query('Whole\_weight\_class=="Less 1"')['Shucked\_weight'],  label='Less 1',alpha =1, c='#66ff33')  ax.scatter(df.query('Whole\_weight\_class=="Over 1"')['Height'],  df.query('Whole\_weight\_class=="Over 1"')['Shucked\_weight'],  label='Over 1', alpha =1, c='#0000ff')  ax.set\_xlabel('Height')  ax.set\_ylabel('Shucked\_weight')  plt.legend(loc='upper left') |
| dat = {'y\_Actual': y\_test, 'y\_Predicted': y\_pred}  dff = pd.DataFrame(dat, columns=['y\_Actual', 'y\_Predicted'])  cross\_table=pd.crosstab(dff['y\_Actual'], dff['y\_Predicted'],  rownames=['Actual'], colnames=['Predicted'], margins=True)  print(cross\_table) |