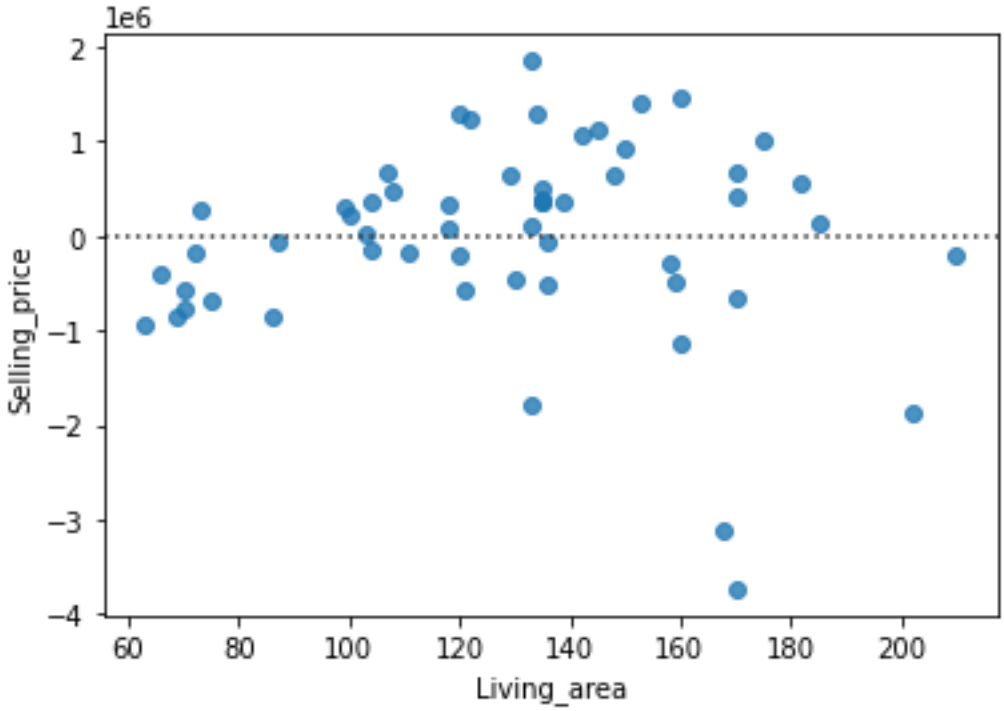
Nikolay Shivarov – Assignment 2

Villas

Firstly, I created a dataframe from the csv file, then in the variable areaPrice I only took the columns “Living_area” and “Selling_price”, because the others are not needed. There were no null values in those two columns, so I didn’t do more data cleaning. Then I created a LinearRegression object, whose first argument is “Living_area”, because we want to predict a price by using the area.

The intercept of the linear regression is 2220603.24 and the slope is 19370.14. Our model predicted that a villa with an area of 100 square metres will be sold for 4157617.1, for 150 square metres the estimated price was 5126124 and for 200 square metres 6094631.

This is the residual plot:



Iris

Firstly, I only took the target and the data from load_iris, the other information is not needed, then I split the data into a training set and a test set to see how effective the model is. The test set is 30% of the whole data. Then I created a logistic regression using the training set. I used the test set to see how accurate the model is. The confusion matrix helped me to see that it was right 35 times and wrong 3 times. This is the confusion matrix:

[[16 0 0]

 [0 7 1]

 [0 2 12]]

Code for the villas:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

import seaborn as sns



## reading the csv

df = pd.read_csv("data_assignment2.csv")

## removing the unnecesary columns

areaPrice = df[["Living_area","Selling_price"]]


## creating a linear regression

LR = LinearRegression()

LR.fit(areaPrice["Living_area"].values.reshape(-1,1),areaPrice["Selling_price"].values.reshape(-
1,1))


## calculating the intercept and the slope

print("The intercept is: ", LR.intercept_[0])

print("The slope is: ", LR.coef_[0][0])


## calculating the estimated price for 100, 150, 200 sq m
```

```
print("Estimated price for 100m^2: " , LR.predict(np.array([[100]]))[0][0])
print("Estimated price for 150m^2: " , LR.predict(np.array([[150]]))[0][0])
print("Estimated price for 200m^2: " , LR.predict(np.array([[200]]))[0][0])

## creating a residual plot
sns.residplot(x='Living_area', y='Selling_price', data=areaPrice)
```

Code for the Iris:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import confusion_matrix

## Loading the data from Iris
iris = load_iris()
data = iris.data
target = iris.target

## spliting the data into a training set and test set
X_train, X_test, Y_train, Y_test = train_test_split(data, target)

## Creating a logistic regression
logistic_model = LogisticRegression()
logistic_model.fit(X_train, Y_train)

## Making a prediction on the test data
```

```
Y_predict = logistic_model.predict(X_test)


## Creating the confusion matrix
confusionMatrix = confusion_matrix(Y_test, Y_predict)
print(confusionMatrix)
```