ML project code-Copy1

March 10, 2022

[2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt    # library providing tools for plotting data
from sklearn.preprocessing import PolynomialFeatures
import seaborn as sns  #data visualization library
from sklearn.linear_model import LinearRegression, HuberRegressor    # classes␣
 ↪providing Linear Regression with ordinary squared error loss and Huber loss,␣
 ↪respectively
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, KFold
```

[3]:
```python
data = pd.read_csv('Athlete_data.csv', sep = ';')
data.head(5)
```

[3]:
```
   year  400m hurdles  200  meters
0  2021          56.72        24.14
1  2021          56.94        24.88
2  2021          58.72        25.11
3  2021          58.92        25.36
4  2021          59.07        25.63
```

[4]:
```python
data2 = data.drop(['year'],axis = 1)

data2.columns = ['400mh','200m']


data2.head(5)
```

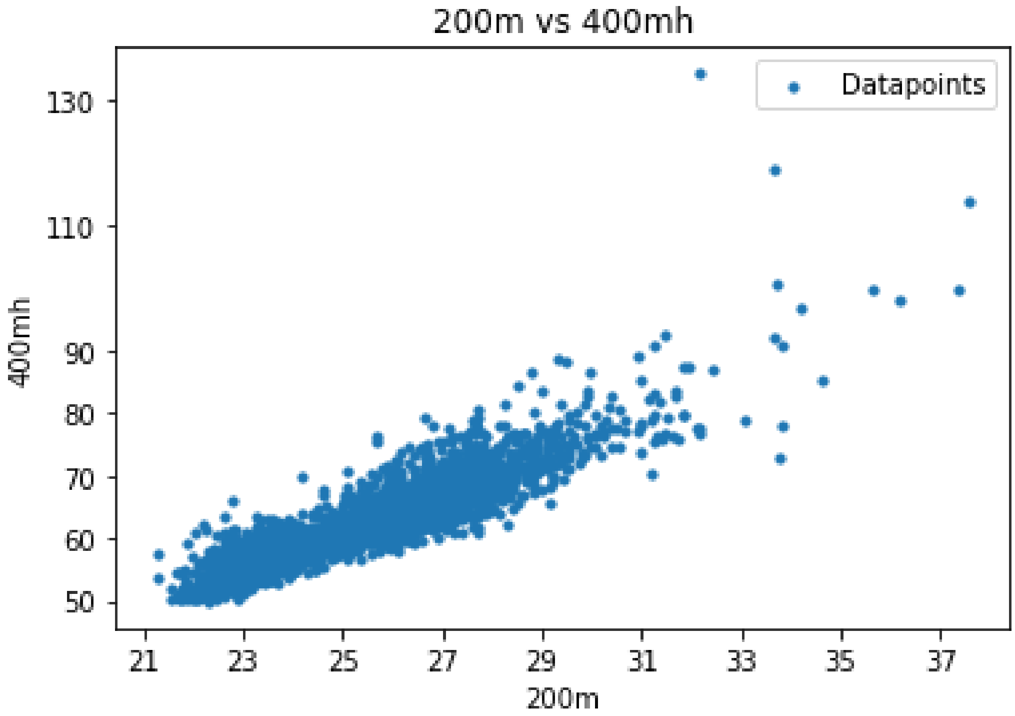
[4]:
```
   400mh   200m
0  56.72  24.14
1  56.94  24.88
2  58.72  25.11
3  58.92  25.36
4  59.07  25.63
```

[5]:
```python
x1 = data2['200m'].to_numpy()
X1 = x1.reshape(-1,1)
```

```
y1 = data2['400mh'].to_numpy()

y1.size
X1.shape
```

```
[5]: (2735, 1)
```

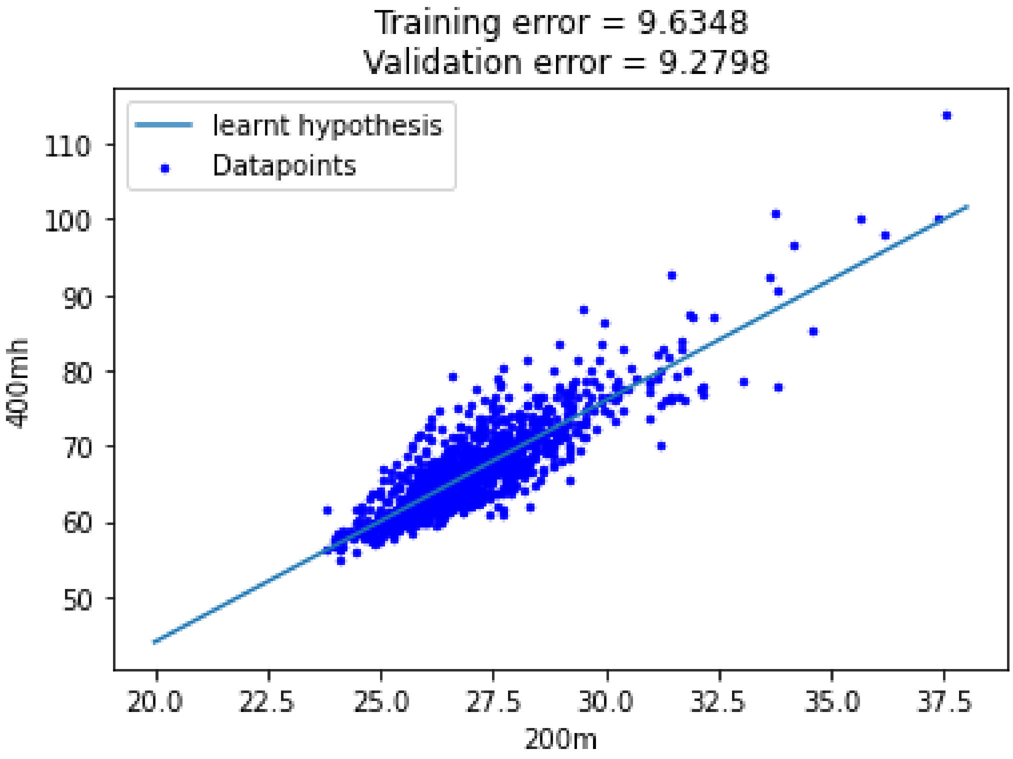
```
[6]: plt.scatter(X1,y1, s=10, label ="Datapoints");
     plt.xlabel("200m")
     plt.xticks([21,23,25,27,29,31,33,35,37])
     plt.ylabel("400mh")
     plt.yticks([50,60,70,80,90,110, 130])
     plt.title("200m vs 400mh")
     plt.legend(loc="best")

     plt.show()
```



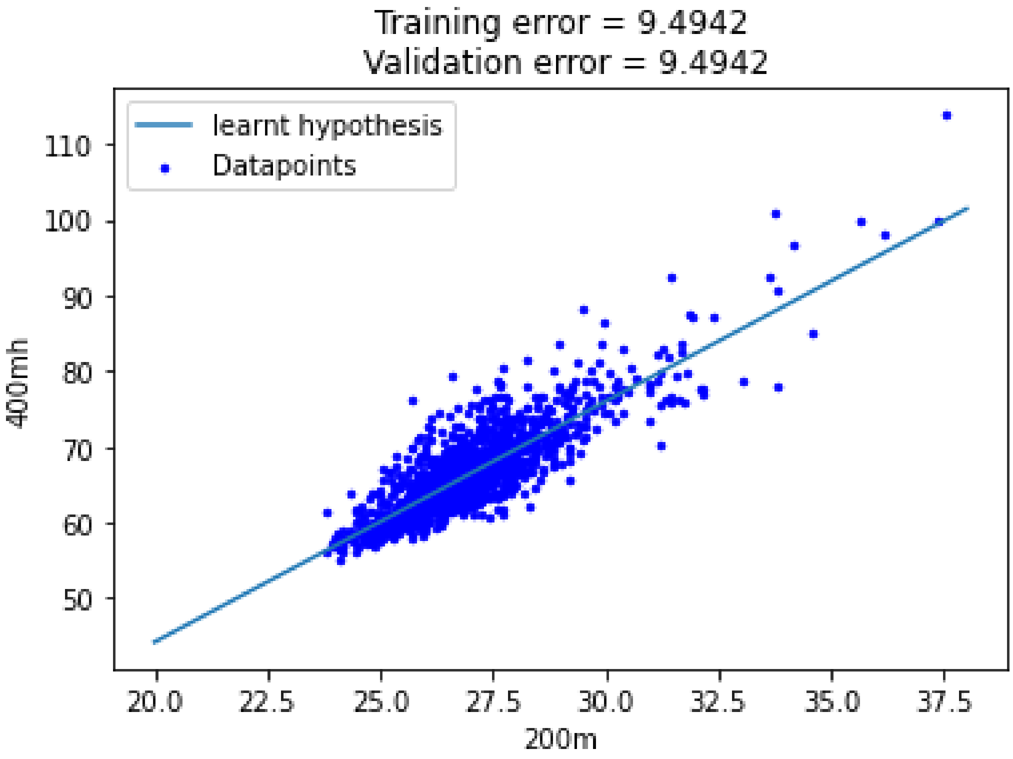
```
[7]: trainingset_size = [0.4, 0.5, 0.6, 0.7]     # set the different sizes of␣
      ↪training set

     for i in range(len(trainingset_size)):     # use for-loop to fit linear␣
      ↪regression models with different sizes of training set
```

```
    index = np.arange(int(len(X1)*trainingset_size[i]))
    print("\nNumber of datapoints in this subset: ",len(index))

    X_sub = X1[index]     # obtain a subset
    y_sub = y1[index]

    # Calculate training error of Huber model
    hmodel = HuberRegressor()
    hmodel.fit(X_sub,y_sub)
    y_pred = hmodel.predict(X_sub)
    tr_error = mean_squared_error(y_sub, y_pred)

    #Calculate validation error
    val_index = np.arange(int(len(X1)*(1-trainingset_size[i])))
    X_val_sub = X1[val_index]
    y_val_sub = y1[val_index]

    y_pred_val_sub = hmodel.predict(X_val_sub)
    val_error_sub = mean_squared_error(y_val_sub,y_pred_val_sub)

    X_fit = np.linspace(20, 38, 100)     # generate samples
    plt.plot(X_fit, hmodel.predict((X_fit.reshape(-1, 1))), label="learnt␣
 ↪hypothesis")    # plot the linear regression model
    plt.scatter(X_sub, y_sub, color = 'b', s=6, label = 'Datapoints')     # plot␣
 ↪a scatter plot of y(200m) vs. X(400mh)
    plt.xlabel('200m')     # set the label for the x/y-axis
    plt.ylabel('400mh')
    plt.legend(loc="best")     # set the location of the legend
    plt.title(f'Training error = {tr_error:.5}\n Validation error =␣
 ↪{val_error_sub:.5}')     # set the title
    plt.show()     # show the plot
```

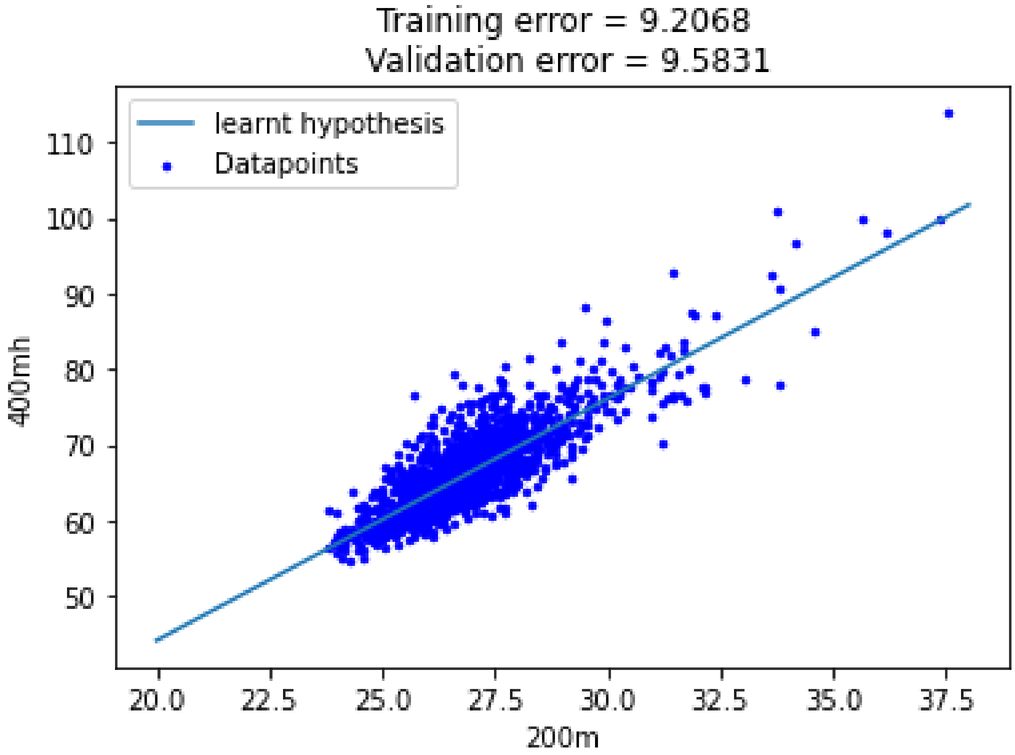
```
Number of datapoints in this subset:  1094
```



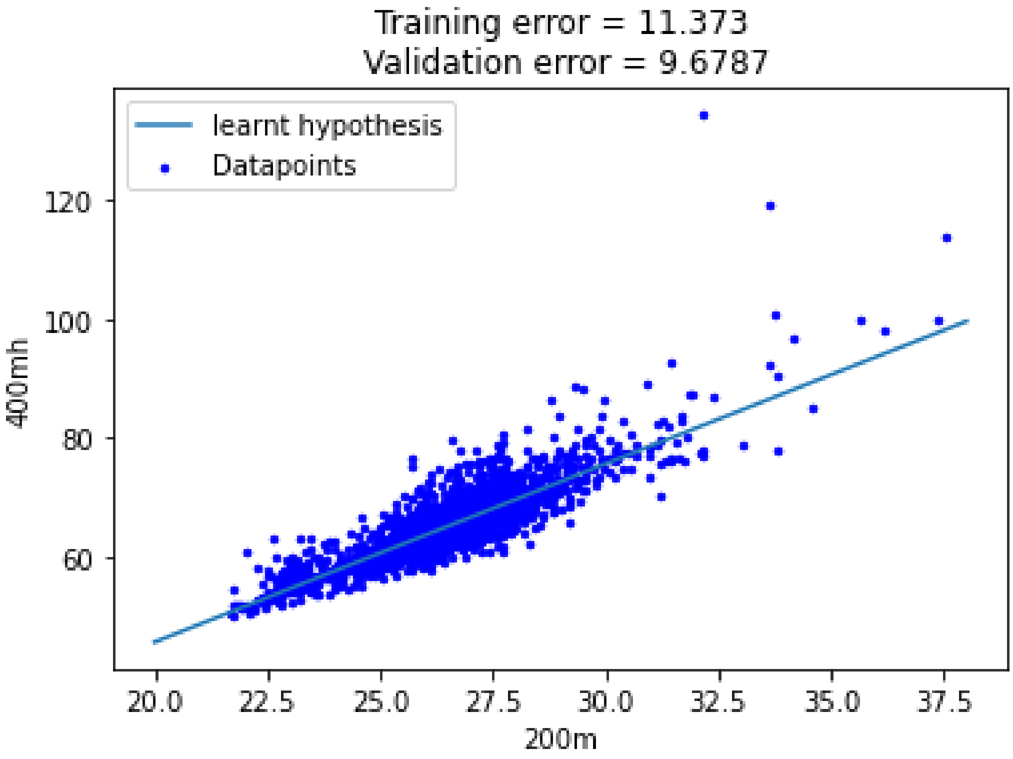
```
Number of datapoints in this subset:  1367
```



```
Number of datapoints in this subset:  1641
```



```
Number of datapoints in this subset:  1914
```



[8]:
```
## Splitting data with 8:2 ratio
X_train, X_test, y_train, y_test = train_test_split(X1,y1, test_size = 0.2,␣
 ↪random_state = 42)
```

[ ]:
```

```

[9]:
```
## Training set evaluated/trained using k-fold cross validation

cv = KFold(n_splits = 10, shuffle = True, random_state = 42)
val_errors2 = []
tr_errors2 = []
# Iterate through the indices of train and validation (iteration through each␣
 ↪split of 20)
for train_index, val_index in cv.split(y_train):

    X_train2, X_val2 = X_train[train_index], X_train[val_index]
    y_train2, y_val2 = y_train[train_index], y_train[val_index]


    hmodel3 = HuberRegressor()
    hmodel3.fit(X_train2, y_train2)# apply Huber regression to these new features and labels
    y_pred_train2 = hmodel3.predict(X_train2)
    tr_error3 = mean_squared_error(y_train2, y_pred_train2)
    print('Training error = {:.5}'.format(tr_error3))
    tr_errors2.append(tr_error3)

    #predict values for the validation data using the linear model
    #calculate the validation error

    y_pred_val2 = hmodel3.predict(X_val2)
    val_error2 = mean_squared_error(y_val2, y_pred_val2)

    print('Validation error= {:.5}'.format(val_error2))
    val_errors2.append(val_error2)
```

```
Training error = 10.617
Validation error= 8.2603
Training error = 10.474
Validation error= 9.8118
Training error = 9.01
Validation error= 23.182
Training error = 10.742
Validation error= 7.8405
Training error = 10.652
Validation error= 8.1054
Training error = 10.434
Validation error= 10.627
Training error = 10.073
Validation error= 13.853
Training error = 10.869
Validation error= 6.23
Training error = 10.554
Validation error= 9.1736
Training error = 10.775
Validation error= 7.276
```

[10]:
```
#Calculate average validation error
sum = 0
for i in val_errors2:
    sum = sum + i

avg_val_error = sum/(len(val_errors2))
print('Average validation error= {:.5}'.format(avg_val_error))
```

```
Average validation error= 10.436
```

[11]:

```
#Calculate average training error
sum2 = 0
for i in tr_errors2:
    sum2 = sum2 +i

avg_train_error = sum2/(len(tr_errors2))
print('Average training error = {:.5}'.format(avg_train_error))
```

```
Average training error = 10.42
```

[ ]:

```

```