ML project code-Copy1

March 10, 2022

[1]:
```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
```

[2]:
```python
data = pd.read_csv('Athlete_data.csv', sep = ';')
data.head(5)
```

[2]:

	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

[3]:
```python
data2 = data.drop(['year'],axis = 1)

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


data2.head(5)
```

[3]:

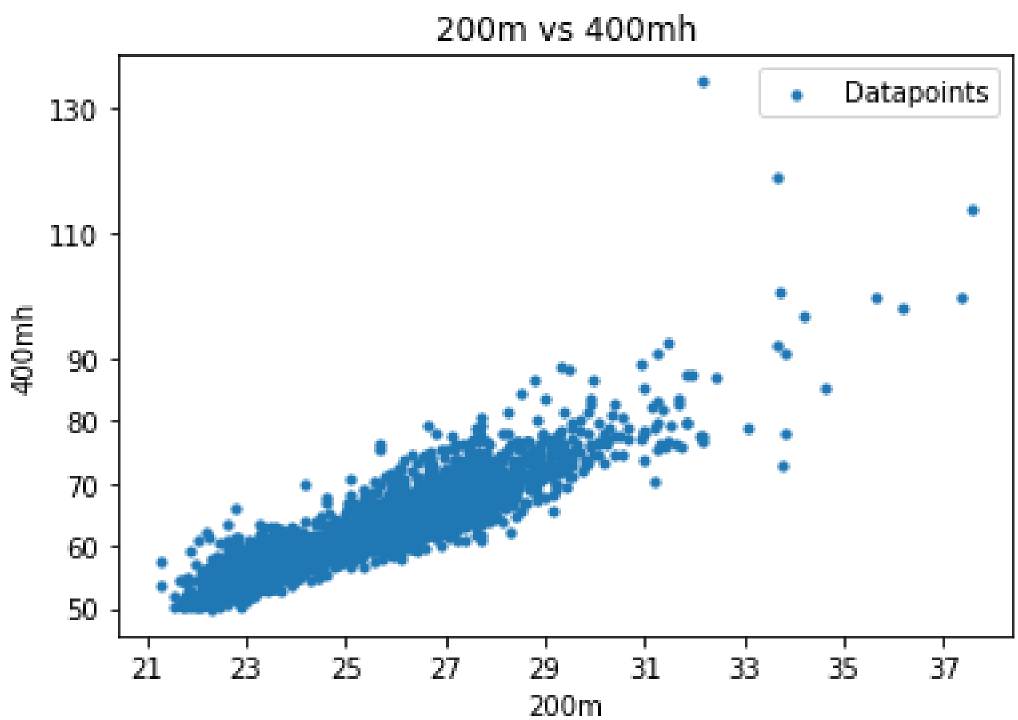
	400mh	200m
0	56.72	24.14
1	56.94	24.88
2	58.72	25.11
3	58.92	25.36
4	59.07	25.63

[4]:
```python
x1 = data2['200m'].to_numpy()
X1 = x1.reshape(-1,1)
```

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

y1.size
X1.shape
```

[4]: (2735, 1)

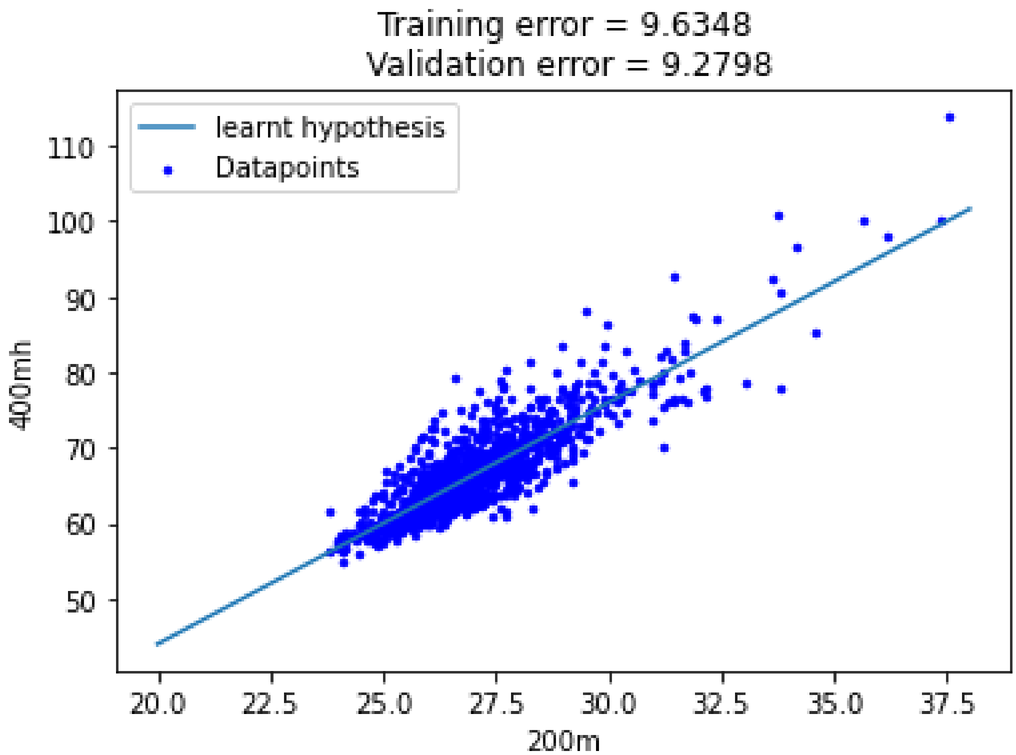
[5]:
```
plt.scatter(X1,y1, s=10, label ="Datapoints");
plt.xlabel("200m")
plt.xticks([21,23,25,27,29,31,33,35,37])
plt.ylabel("400mh")
plt.yticks([50,60,70,80,90,110, 130])
plt.title("200m vs 400mh")
plt.legend(loc="best")

plt.show()
```



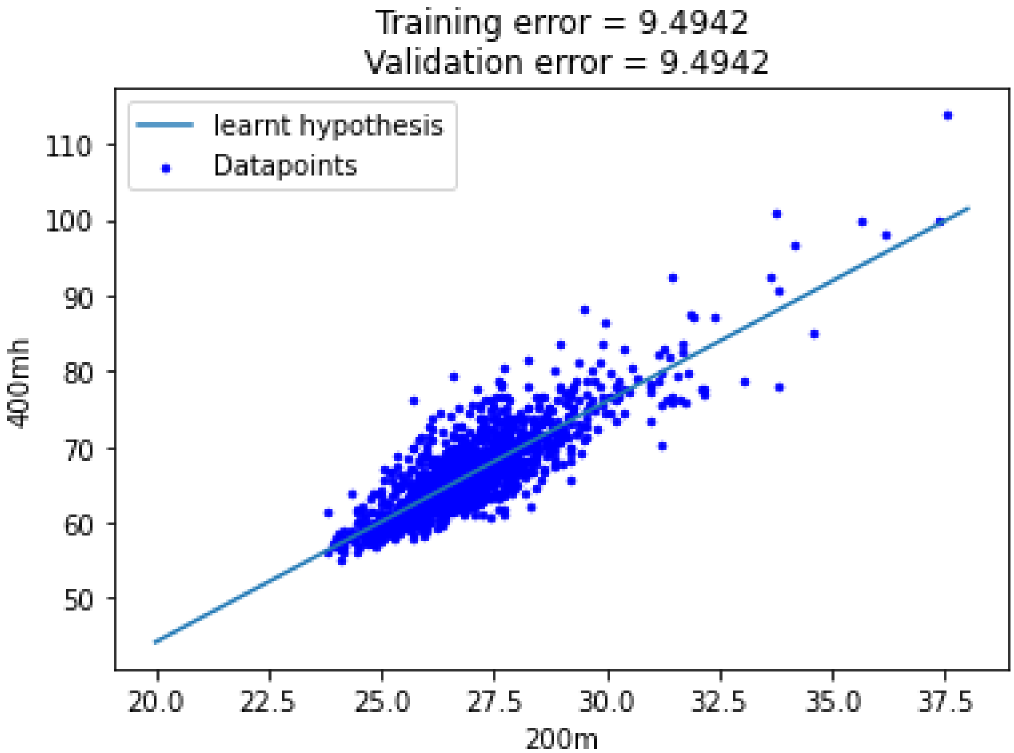
[13]:
```
trainingset_size = [0.4, 0.5, 0.6, 0.7]     # set the different sizes of␣
↪training set

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, NOTE: this is the subset of␣
↪features you will use for this task
    y_sub = y1[index]

    # Calculate training error of linear 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
```

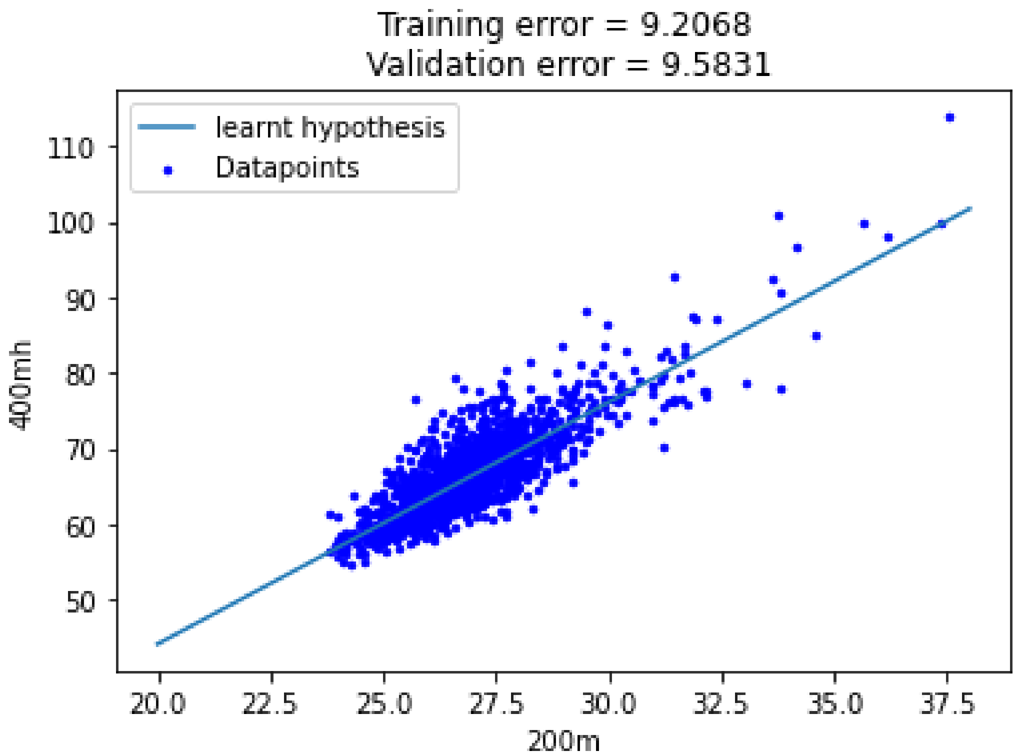
```
Number of datapoints in this subset:  1094
```



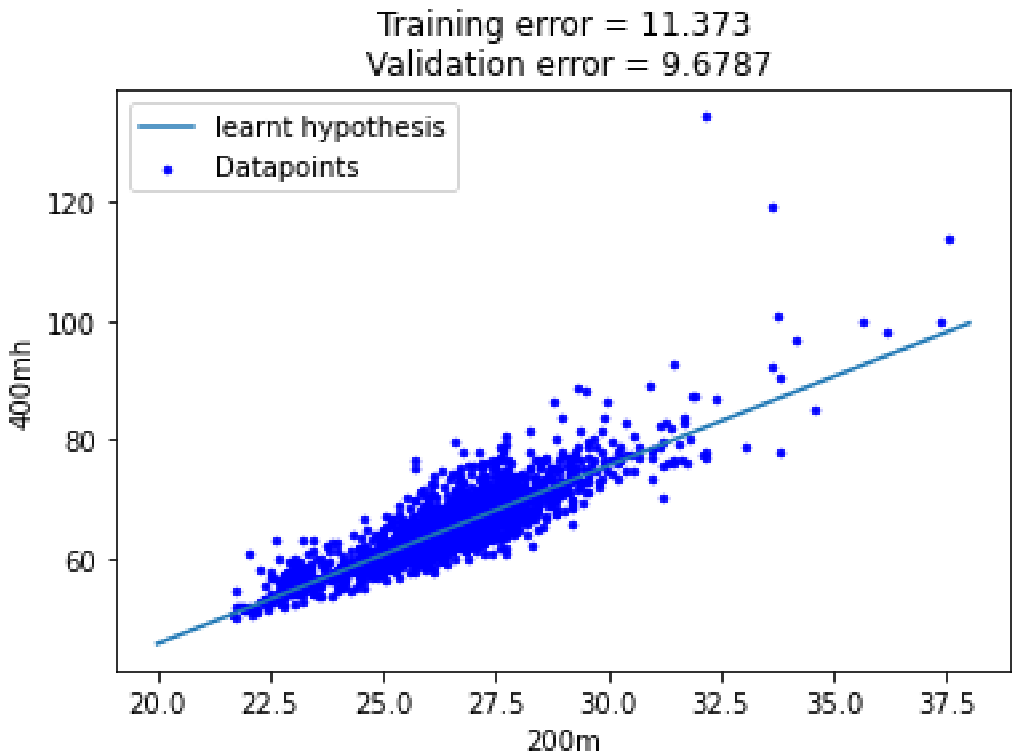

```
Number of datapoints in this subset:  1367
```




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




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



[7]:

```
## Trying out train_test_split with 5:3:2
X_train, X_rem, y_train, y_rem = train_test_split(X1,y1, test_size = 0.5,␣
 ↪random_state = 42)
X_val,X_test,y_val,y_test = train_test_split(X_rem,y_rem,test_size = 0.2,␣
 ↪random_state = 42)

hmodel2 = HuberRegressor()
hmodel2.fit(X_train,y_train)
y_pred_train = hmodel2.predict(X_train)
tr_error2 = mean_squared_error(y_train, y_pred_train)

y_pred_val = hmodel2.predict(X_val)
val_error = mean_squared_error(y_val, y_pred_val)

print('Training error= {:.5}'.format(tr_error2))
```

```
Training error= 10.999
```

[14]:

```
print('Validation error={:.5}'.format(val_error))

#test error
```

```python
y_pred_test = hmodel2.predict(X_test)
test_error = mean_squared_error(y_test, y_pred_test)
print('Test error = {:.5}'.format(test_error))
```

```
Validation error=9.6298
Test error = 8.7665
```

[9]:
```python
## Trying out splitting data into training/validating through 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(y1):

    X_train2, X_val2 = X1[train_index], X1[val_index]
    y_train2, y_val2 = y1[train_index], y1[val_index]


    hmodel3 = HuberRegressor()

    hmodel3.fit(X_train2, y_train2)# apply linear 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 = 10.247
Validation error= 8.0741
Training error = 10.152
Validation error= 9.1336
Training error = 9.9104
Validation error= 11.291
Training error = 10.215
```

```
Validation error= 8.3473
Training error = 10.113
Validation error= 9.2733
Training error = 10.096
Validation error= 9.7091
Training error = 10.197
Validation error= 8.0873
Training error = 10.176
Validation error= 8.378
Training error = 10.044
Validation error= 9.453
Training error = 9.0301
Validation error= 18.637
```

[10]:

```python
#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= 10.038
```

[11]:

```python
#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 = 10.018
```

[ ]:

```python

```