ML project code-Copy1

March 31, 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
## Most code is from the CS-C3240 assignments with slight changes
```

[4]:
```python
data = pd.read_csv('Athlete_data.csv', sep = ';')
```

[5]:
```python
data.columns = ['year', '400mh SB', '200m SB']

data.head(5)
```

[5]:

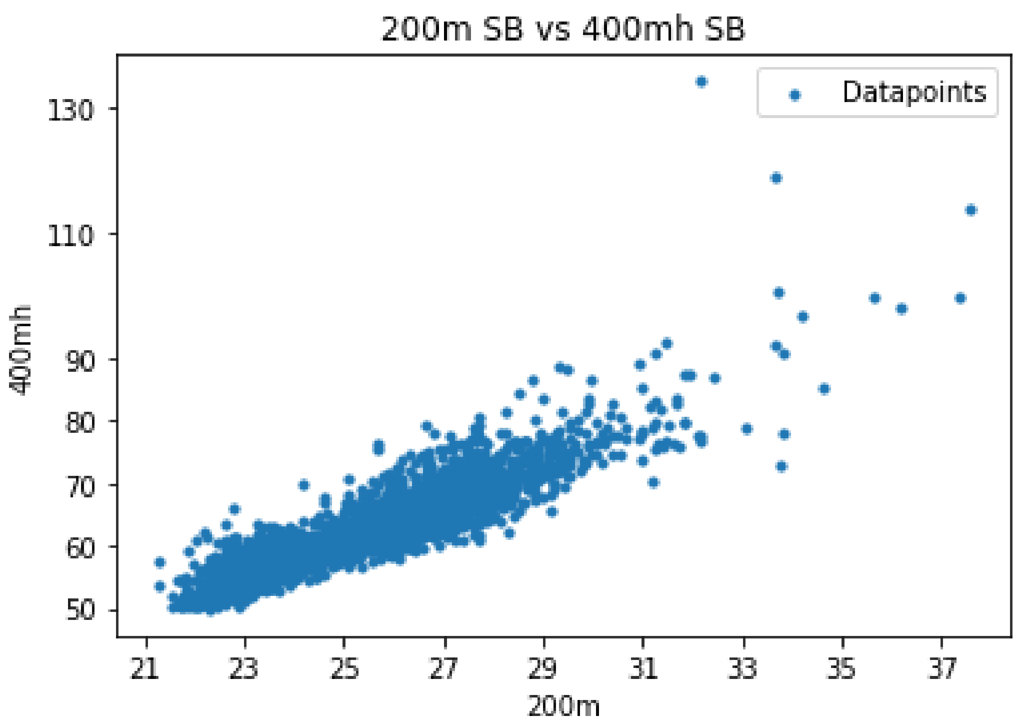
	year	400mh SB	200m SB
0	2021	56.72	24.14
1	2021	56.94	24.88
2	2021	58.72	25.11
3	2021	58.92	25.36
4	2021	59.07	25.63

[6]:
```python
X1 = data['200m SB'].to_numpy().reshape(-1,1)
X2 = data['year'].to_numpy().reshape(-1,1)

y1 = data['400mh SB'].to_numpy()

y1.size
```

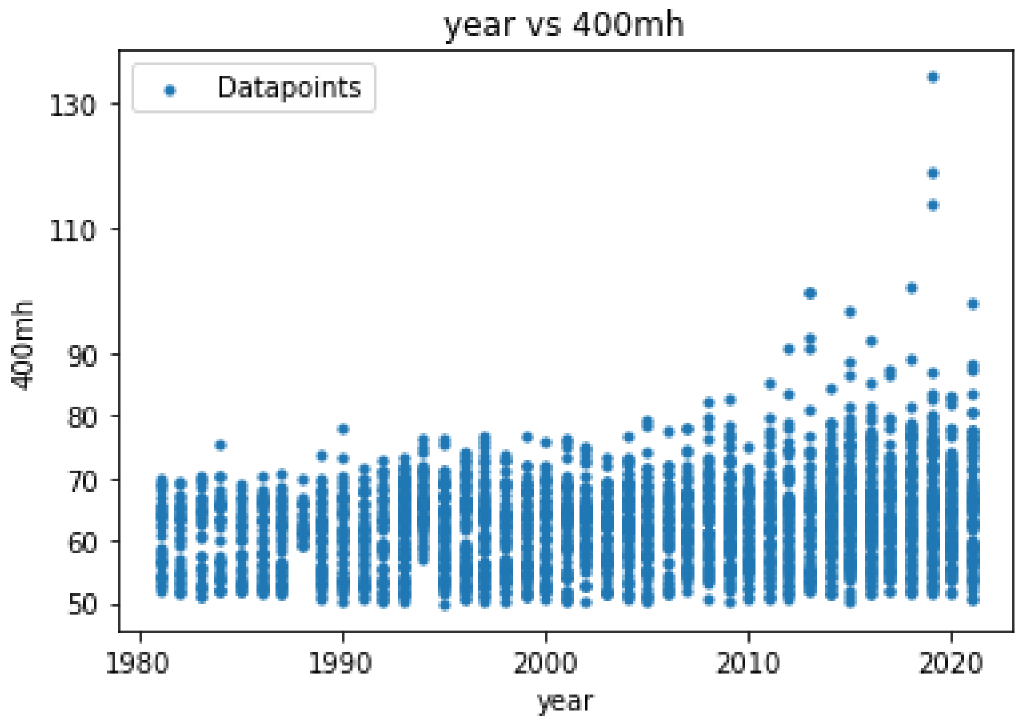
[6]: 2735

[7]:
```python
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 SB vs 400mh SB")
plt.legend(loc="best")

plt.show()
```



[8]:
```
plt.scatter(X2,y1, s=10, label ="Datapoints");
plt.xlabel("year")
plt.xticks([1980,1990,2000,2010,2020])
plt.ylabel("400mh")
plt.yticks([50,60,70,80,90,110, 130])
plt.title("year vs 400mh")
plt.legend(loc="best")

plt.show()
```



[9]:
```
## Splitting data with 8:2 ratio, 0.2 for testing
X_train, X_test, y_train, y_test = train_test_split(X1,y1, test_size = 0.2,␣
↪random_state = 42)

## X_test and y_test are left to wait for final test error
```

[28]:
```
## Training set evaluated/trained using k-fold cross validation, Huber␣
↪Regression

cv = KFold(n_splits = 10, shuffle = True, random_state = 42)
val_errors = []
tr_errors = []
# 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(epsilon = 1.9)
```

```
    hmodel3.fit(X_train2, y_train2)# apply Huber regression to these new␣
↪features and labels
    y_pred_train2 = hmodel3.predict(X_train2)
    tr_error = mean_squared_error(y_train2, y_pred_train2)
    tr_errors.append(tr_error)

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

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

    val_errors.append(val_error)
```

[29]:
```
#Calculate average validation error
sum = 0
for i in val_errors:
    sum = sum + i

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

```
Average validation error= 10.344
```

[30]:
```
#Calculate average training error
sum2 = 0
for i in tr_errors:
    sum2 = sum2 +i

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

```
Average training error = 10.328
```

[31]:
```
## Training set evaluated/trained with linear regressor 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_train3, X_val3 = X_train[train_index], X_train[val_index]
```

```
    y_train3, y_val3 = y_train[train_index], y_train[val_index]


    lin_regr = LinearRegression()

    lin_regr.fit(X_train3, y_train3)# apply Linear regression to these new␣
↪features and labels
    y_pred_train3 = lin_regr.predict(X_train3)
    tr_error2 = mean_squared_error(y_train3, y_pred_train3)
    tr_errors2.append(tr_error2)


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

    y_pred_val3 = lin_regr.predict(X_val3)
    val_error2 = mean_squared_error(y_val3, y_pred_val3)

    val_errors2.append(val_error2)
```

[32]:

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

avg_val_error2 = sum/(len(val_errors2))
print('Average validation error= {:.5}'.format(avg_val_error2))
```

```
Average validation error= 10.279
```

[33]:

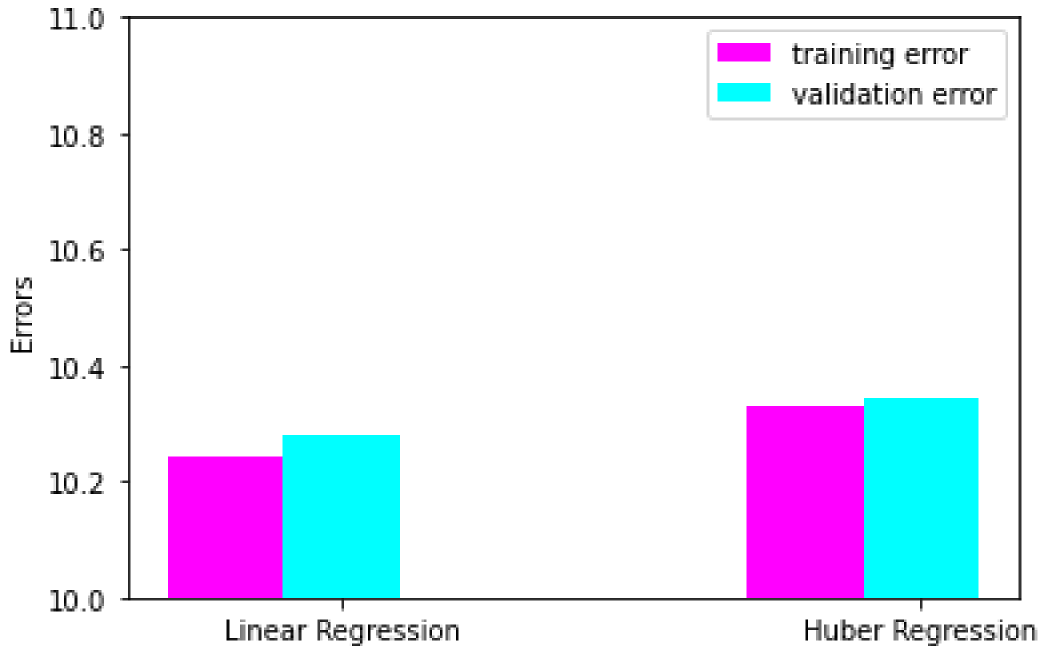
```
#Calculate average training error
sum2 = 0
for i in tr_errors2:
    sum2 = sum2 +i

avg_train_error2 = sum2/(len(tr_errors2))
print('Average training error = {:.5}'.format(avg_train_error2))
```

```
Average training error = 10.242
```

[34]:

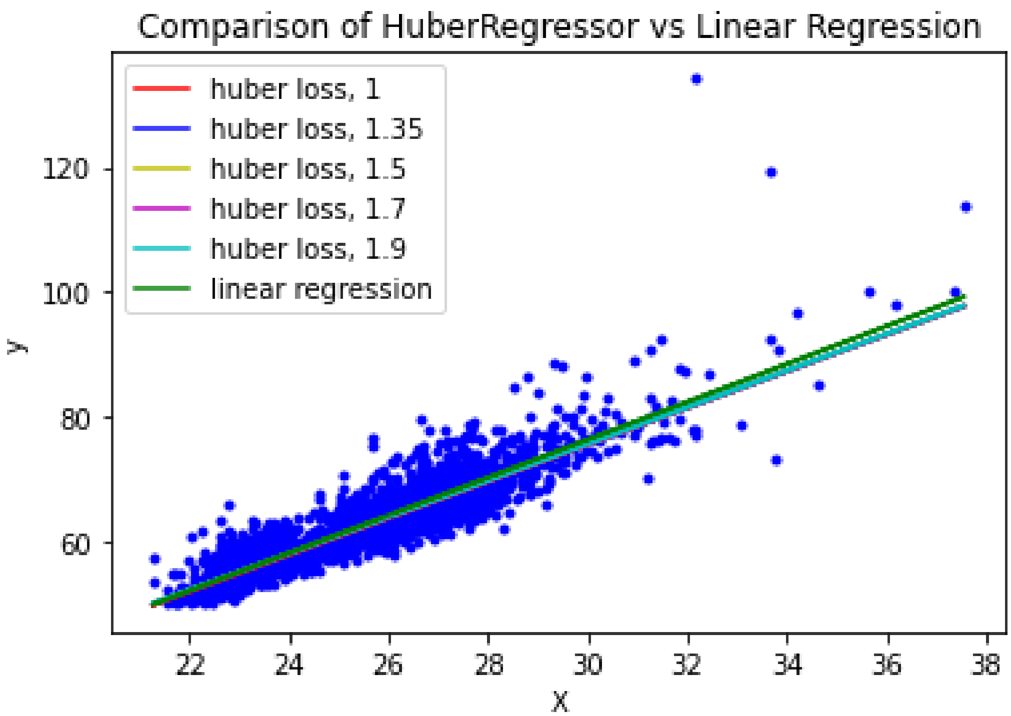
```
x = np.arange(2)
y_1 = [avg_train_error2,avg_train_error]
y_2 = [avg_val_error2, avg_val_error]
width = 0.2

plt.bar(x-0.2, y_1, width, color='magenta')
```

```
plt.bar(x, y_2, width, color='cyan')
plt.ylim(10, 11)
plt.xticks(x,['Linear Regression', 'Huber Regression'] )
plt.ylabel("Errors")
plt.legend(["training error", "validation error"])
plt.show()
```



[37]:
```
plt.plot(X_train, y_train, "b.")

# Fit the huber regressor over a series of epsilon values.
colors = ["r-", "b-", "y-", "m-", "c-"]
r2_scores = []

epsilon_values = [1,1.35, 1.5, 1.7, 1.9]
for k, epsilon in enumerate(epsilon_values):
    huber = HuberRegressor(alpha=0.0, epsilon=epsilon)
    huber.fit(X_train, y_train)
    coef_ = huber.coef_ * X_train + huber.intercept_
    r2_scores.append(huber.score(X_train,y_train))

    plt.plot(X_train, coef_, colors[k], label="huber loss, %s" % epsilon)

# Fit a linear regressor to compare it to huber regressor.
linear = LinearRegression()
linear.fit(X_train, y_train)
```

```
coef_linear = linear.coef_
coef_ = linear.coef_ * X_train + linear.intercept_
plt.plot(X_train, coef_, "g-", label="linear regression")

plt.title("Comparison of HuberRegressor vs Linear Regression")
plt.xlabel("X")
plt.ylabel("y")
plt.legend(loc=0)
plt.show()
```



[38]:
```
for i, epsilon in enumerate(epsilon_values):
    print('Huber score for epsilon value {:1} = {:.5}'.format(epsilon,␣
 ↪r2_scores[i]))
```

```
Huber score for epsilon value 1 = 0.8125
Huber score for epsilon value 1.35 = 0.81353
Huber score for epsilon value 1.5 = 0.81418
Huber score for epsilon value 1.7 = 0.81475
Huber score for epsilon value 1.9 = 0.81517
```

[40]:
```
## Final test error:

linear_reg = LinearRegression()
```

```python
linear_reg.fit(X_test, y_test)
y_pred_test = linear_reg.predict(X_test)

test_error = mean_squared_error(y_pred_test, y_test)

print("The test error is: {:.3}".format(test_error))
```

```
The test error is: 8.33
```

[ ]:

```python

```