Recipe Name: read_matrix

Inputs:

input_string, a string with data

Outputs:

data_matrix, a matrix filled with data from *input_string*

Steps:

1. Let *sequence_matrix* be an empty sequence
2. Let *inner_sequence* be an empty sequence
3. For each line in *input_string*, do the following:
    a. *inner_sequence* ← a sequence formed by taking each value from the line, with separation between elements occurring by comma and whitespace
    b. Append *inner_sequence* to the end of *sequence_matrix*
4. *data_matrix* ← a matrix formed by converting *sequence_matrix* to a matrix, where each element of *sequence_matrix* is a row in *data_matrix*
5. Return *data_matrix*

Recipe Name: generate_predictions

Inputs:

inputs, an n * m matrix of variables

weights, an m * 1 matrix representing the linear model

Outputs:

an n*1 matrix of values predicted by the model using the data *inputs*

Steps:

1. Return the result of using matrix multiplication to multiply *inputs* by *weights*

Recipe Name: prediction_error

Inputs:

model, a linear model using an m * 1 matrix of weights

inputs, an n * m matrix of variables

actual, an n * 1 matrix of the actual result of *inputs*

Outputs:

The mean squared error between the values predicted by *model* and *actual*

Steps:

1. *generated* ← generate_predictions(*model*, *inputs*)
2. *actual_seq* ← *actual* converted to a sequence by setting each row of *actual* to an individual element of *actual_seq*
3. *generated_seq* ← *generated* converted to a sequence by setting each row of *actual* to an individual element of *generated_seq*
4. Return mse(*actual_seq*, *generated_seq*)

Derivation

$$\frac{dMSE(w)}{dw} = \frac{1}{n}(2w^T X^T X - 2y^T X) = 0$$

$$2w^T X^T X - 2y^T X = 0$$

$$= w^T X^T X - y^T X$$

$$= (w^T X^T X - y^T X)^T$$

$$= X^T X w - X^T y$$

$$\Rightarrow X^T X w = X^T y$$

$$\Rightarrow (X^T X)^{-1} X^T X w = (X^T X)^{-1} X^T y$$

$$=> w = (X^T X)^{-1} X^T y$$

Codeskulptor Link

https://py3.codeskulptor.org/#user305_MI244HQ3wp_27.py

Baseball Performance Prediction Errors

500 iterations,

```
1954-2000 data error:
Least squares: 93.73996861983817
LASSO, lambda = 5000: 128.9135542819172
LASSO, lambda = 10000: 133.4602047909331
LASSO, lambda = 30000: 137.8306675908317

2001-2012 data error using model of 1954-2000:
Least squares: 105.0855290667178
LASSO, lambda = 5000: 95.95344487062134
LASSO, lambda = 10000: 94.47655073093254
LASSO, lambda = 30000: 92.40219582521325
```

Discussion

1. This expression is true because of the properties of transposing a composition of matrices and because each matrix w, X, and y had dimensions that worked to allow for the composition to take place.
2. An increase in λ means that the 1-norm of the weights must decrease to match, which means the summation of the absolute value of each weight must decrease in turn.
3. From the training data, least squares produced the lowest MSE. When used on the test 2001-2012 data, it was actually the LASSO with the highest lambda value that minimized MSE. It seems like LASSO is better at predicting future data than the actual values of the training data.
4. It seems like Hits and Earned Runs are less important, since the weights on those statistics are significantly more deviated from 0 than the other stats.