Christopher Wang
CSCI 6364 – Machine Learning
George Washington University
Github:

Question 1

a) Suppose you had to predict y without knowledge of x. What value would you predict? What would be it's mean squared error (MSE) on thse four points? (1,1) (1,3) (4,4) (4,6)

To find the y without knowledge of x we'd want to find the best fitting line (regression line) that is the best prediction for the current dataset we currently have without knowledge of x within the formula mx + b. B would be our bias and is the avg of all the y values together which is 3.5 would be the value we predict. Our mean squared error would be (2.5^2+0.5^2+0.5^2+2.5^2)/4 = 3.25

b) Now let's say you want to predict y based on x. What is the MSE of the linear function y = ax on these four points?

A = (summation x*y from 1 to n) / summation x^2 from 1 to n
x*y-bar = 11
x^2-bar =8.5

a = 11/8.5 = ~1.294

y = 1.294x

X value	Y value	Y predicted value	Error Squared
1	1	1.294	0.086
1	3	1.294	2.910
4	4	5.176	1.383
4	6	5.176	0.679

MSE = (0.086+2.910+1.383+0.679) / 4 = 1.265

c) Find the line y = ax + b that minimizes the MSE on these points. What is its MSE?
X bar = 1+1+4+4 / 4 = 10/4 = 5/2
Y bar = 1+3+4+6 / 4 = 14/4 = 7/2
XY bar = (1x1 + 1x3 + 4x4 + 4x6) / 4 = 44/4 = 11
X^2 bar = 1^2 + 1^2 + 4^2 + 4^2 / 4 = 34/4 = 8.5
m = (11-(5/2)(7/2)) / (8.5 - (5/2)^2) = 1
b = y bar - m* xbar = 7/2 - (1)(5/2) = 2/2 = 1
y = mx + b = x + 1

X value	Y value	Y predicted value	Error Squared
1	1	2	1

1	3	2	1
4	4	5	1
4	6	5	1

MSE = (1+1+1+1)/4 = 1

Question 2

The best single indicator for salary is RBIs with a RMSE of 846.042 with the number of hits being the best second indicator with RMSE of 929.539. I started off with plotting the values with salary as our y value and whatever feature as our x value. Then I also created a confusion matrix to see which features best fit with the salary. After verifying which features best correlated I created a linear regression model then outputting the mean squared error and root mean square error to find which among the 6 best correlated features outputted the most minimum value

RBI

```
[28.16376497]
[[27.96950674]]
MSE: 715786.844179358
RMSE 846.0418690463008
```

Runs

```
[-74.25726546]
[[28.3411574]]
MSE: 871448.2541882718
RMSE 933.5139282240366
```

Hits

```
[-165.63352011]
[[15.3642804]]
MSE: 864043.177815198
RMSE 929.5392287661656
```

Doubles

```
[101.36391975]
[[69.82707947]]
MSE: 876971.7164104498
RMSE 936.467680387556
```

Home Runs

```
[101.36391975]
[[69.82707947]]
MSE: 876971.7164104498
RMSE 936.467680387556
```

Walks

```
[256.99258715]
[[28.37357521]]
MSE: 931885.174079028
RMSE 965.3419985057254
```

Question 3

I initially started with having the linear regression model having a tolerance of 0.001 with a max iteration of 10000. This took an extensive amount of time to train based on the training data since it’s such a large data set and the jobs are not being done in parallel. Eventually I put the max iteration down to 1000 and run 5 jobs in parallel in order to cut down on the training time. This took about 13.3 minutes giving us a ~0.9407 accuracy score for the training data, but for the test set it gave us a 0.9171 accuracy score. I used a f-score metric to drill further down into the details giving us the precision, recall, and f1-score to all the classes. The precision stat is an indication of how many true positives is among all the positives results given, while the recall stat tells us the number of true positives over the number of all the samples that were returned as positive. Diving in we can see that the model works over a majority of the numbers, but the classes that give the model the most issues are 3, 5, and 8 being the lowest f1-score. (Most likely due to how similar they are in structure) I then went ahead and tried testing the model with a high tolerance level of 0.01 and as expected the accuracy goes down a bit to ~0.9147 for training data accuracy and ~0.9064 for testing data accuracy.
Comparing our f1-scores to the Kneighbors model we do slightly worse as the K neighbors is able to distinguish between our problem numbers better than our linear regression. I then decided to test to see if we lowered our tolerance level to see if this would improve on our model with the risk of the model

over fitting to our training data. The result is the model eventually hit our max iteration ending in 23 minutes and checking the f-score of the different classes the accuracy didn't improve too much. We can increase the max iteration, but this comes with the expectation that eventually the model will be over fitting to the training data.

Model with 0.001 tolerance:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	1033
1	0.93	0.98	0.95	1171
2	0.93	0.90	0.92	1044
3	0.90	0.89	0.90	1088
4	0.91	0.92	0.92	1018
5	0.87	0.86	0.87	949
6	0.95	0.96	0.95	1034
7	0.93	0.94	0.93	1100
8	0.89	0.84	0.87	1016
9	0.90	0.90	0.90	1047
accuracy			0.92	10500
macro avg	0.92	0.92	0.92	10500
weighted avg	0.92	0.92	0.92	10500

Model with 0.01 tolerance:

	precision	recall	f1-score	support
0	0.95	0.96	0.95	1033
1	0.90	0.98	0.93	1171
2	0.93	0.89	0.91	1044
3	0.90	0.88	0.89	1088
4	0.88	0.92	0.90	1018
5	0.88	0.85	0.86	949
6	0.94	0.96	0.95	1034
7	0.92	0.92	0.92	1100
8	0.88	0.82	0.85	1016
9	0.88	0.87	0.88	1047
accuracy			0.91	10500
macro avg	0.91	0.90	0.91	10500
weighted avg	0.91	0.91	0.91	10500

Kneighbors report:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	1033
1	0.95	0.99	0.97	1171
2	0.95	0.94	0.94	1044
3	0.91	0.93	0.92	1088
4	0.95	0.92	0.93	1018
5	0.92	0.91	0.92	949
6	0.95	0.98	0.96	1034
7	0.93	0.93	0.93	1100
8	0.97	0.87	0.92	1016
9	0.90	0.93	0.92	1047
accuracy			0.94	10500

Model with 0.0001 tolerance:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	1033
1	0.93	0.98	0.95	1171
2	0.93	0.90	0.92	1044
3	0.91	0.88	0.90	1088
4	0.91	0.92	0.92	1018
5	0.87	0.86	0.86	949
6	0.95	0.96	0.96	1034
7	0.93	0.94	0.94	1100
8	0.89	0.84	0.87	1016
9	0.90	0.90	0.90	1047
accuracy			0.92	10500
macro avg	0.92	0.92	0.92	10500
weighted avg	0.92	0.92	0.92	10500