**ECE521 Assignment 1**

Feb 2nd, 2018

Team Members:

Yilin Chen 1000311281

Wenyu Mao 1000822292

Jiaying Wang 1000337502

**Table of Contents**

1. **Contributions**……………………………………………………………..1
2. **Euclidean Distance Function**…………………………………………….1
3. **Making Predictions for Regression**……………………………....……...2
4. **Making Predictions for Classifications**………………………………….7
5. **Contributions**

Yilin Chen: 33.3%

Wenyu Mao: 33.3%

Jiaying Wang: 33.3%

1. **Euclidean distance function**

By observing the equation and result matrix for the Euclidean distance provided in the assignment handout (Figure 2.1), we found that for two matrices, X (N1 \* D) and Z(N2 \* D), the Euclidean distance between them is an N1 \* N2 pairwise squared matrix.

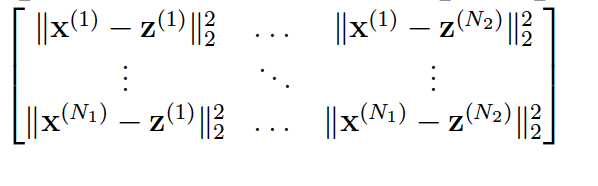


Figure 2.1 Euclidean distance

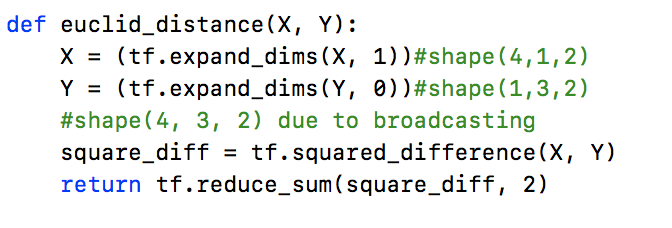


Figure 2.2 Euclidean Distance Function

The solution is shown in Figure 2.2. tf.squared\_difference results in a pairwise matrix between two matrices. It follows TensorFlow broadcasting, which will automatically match the dimensions for two matrices. In order to use this functionality, we first expand the dimensions for two matrices. Then reduce the extra dimension and return the result.

1. **Making Predictions for Regression**

**3.1 Choosing the nearest neighbours**

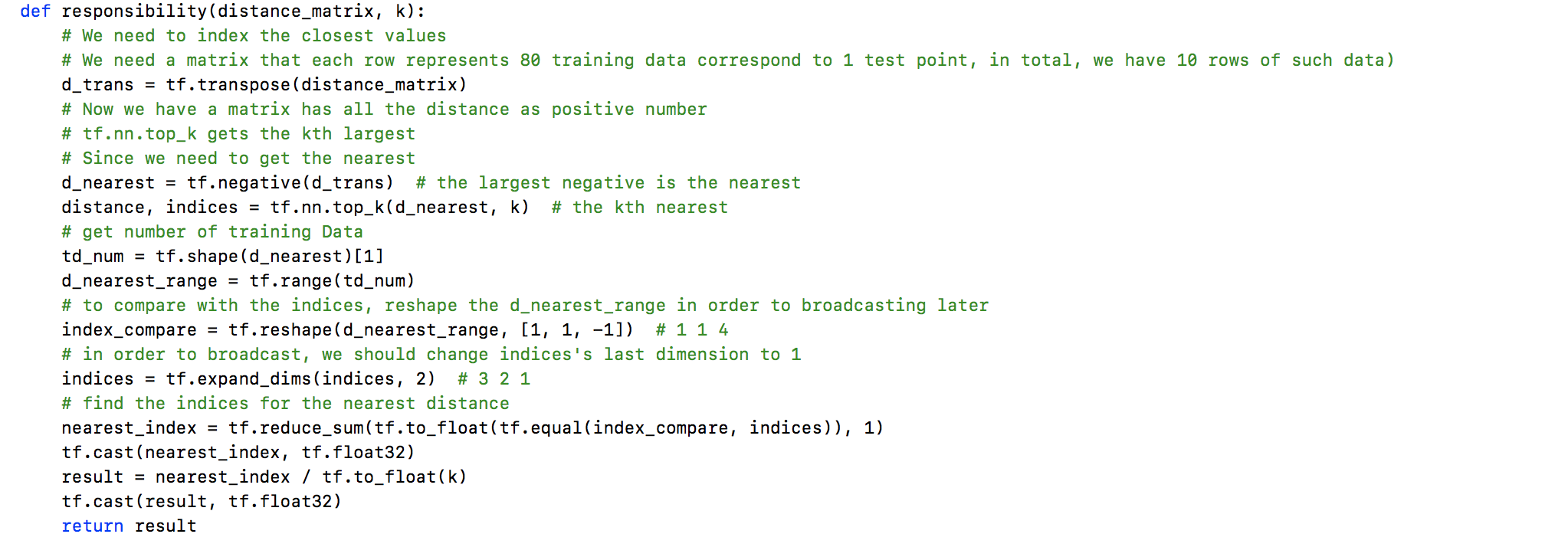
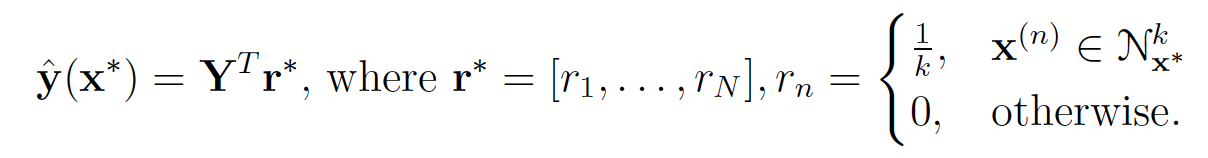


Figure 3.1 Responsibility Function

tf.nn.top\_k function is used to find values and indices of the k largest entries for the last dimension. Since the order of the input arguments of the euclid\_distance function matters and we use X = trainData and Y = testData, the resulted distance\_matrix has to be transposed first to obtain the correct matrix shape (N2 \* N1, N1 is the number of train data, N2 is the number of test data) before using tf.nn.top\_k function. Then the elements of the distance matrix are changed to negative since tf.nn.top\_k function find the largest entries. The nearest data should have the largest values after negation. After the data processing, tf.nn.top\_k function is used to obtain the indices of the k nearest data in the train dataset for each test data. The resulted indices matrix is then expanded and compared with a matrix of all the possible training data indices in order to get a vector like [R1, R2, … Rn] where Ri = 1 if the i-th training data belongs the the neighbor of the closest k samples and Ri = 0 otherwise. The above vector is then divided by k following the formula in the handout.

**3.2 Prediction**



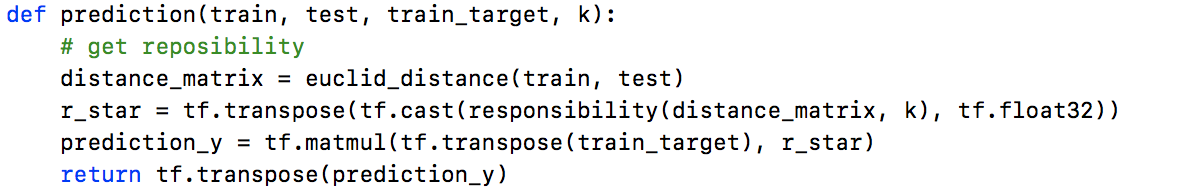
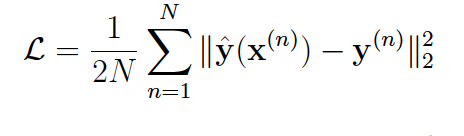
The formula above is used in the prediction function to calculate the predicted results.

Figure 3.2 Prediction function

**3.3 MSE Loss**

****

The formula above is used to calculate the MSE loss. Figure 3.3 shows the implementation function.

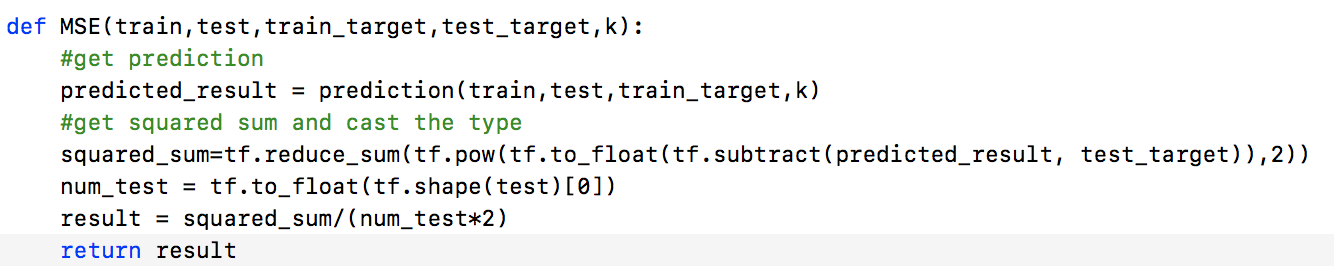


Figure 3.3 MSE function

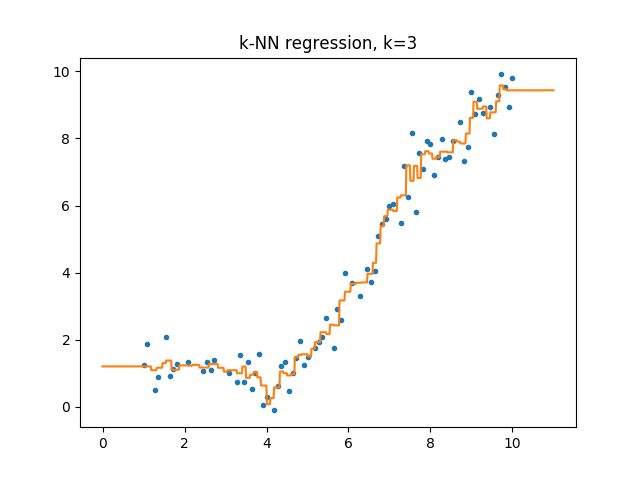
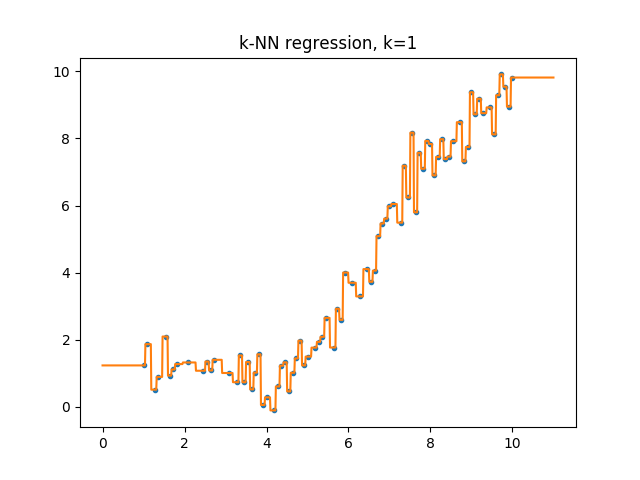
Tables below shows the results. The lowest validation MSE losses is found to occur with k = 1.

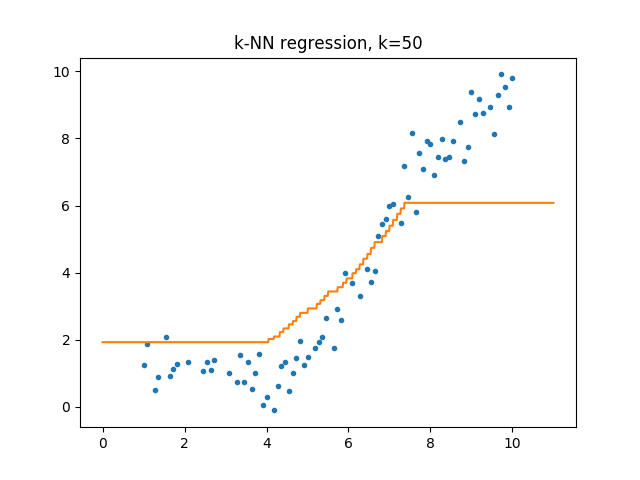
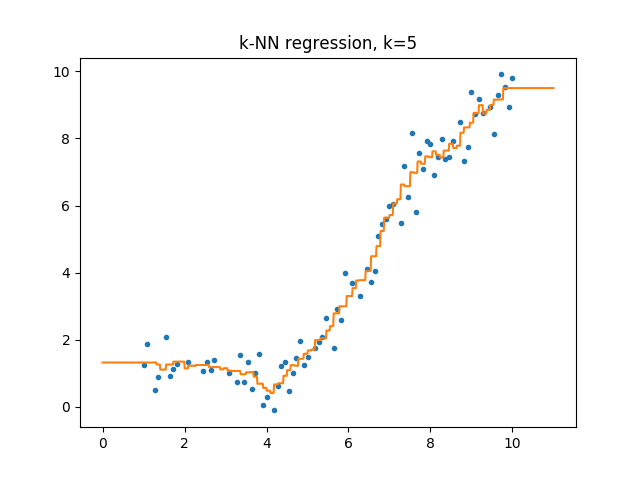
Note: The MSE losses vary when different data type (float32 and float64) are used. But the best k is k = 1 in both cases.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: MSE Losses** (using float32) | | | |
|  | Training MSE Loss | Validation MSE Loss | Testing MSE Loss |
| k=1 | 0.000000 | **0.271550** | 0.311004 |
| k=3 | 0.105241 | 0.326278 | 0.145092 |
| k=5 | 0.118541 | 0.310439 | 0.178327 |
| k=50 | 1.248009 | 1.228702 | 0.707935 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 2: MSE Losses** (using float64) | | | |
|  | Training MSE Loss | Validation MSE Loss | Testing MSE Loss |
| k=1 | 0.000000 | **0.271550** | 0.139432 |
| k=3 | 0.106465 | 0.324376 | 0.158797 |
| k=5 | 0.121068 | 0.316699 | 0.185096 |
| k=50 | 1.245985 | 1.228702 | 0.702632 |

The following figures show the prediction results using different k. At k = 1, the predictions follow the true values exactly with sharp jumps across data points. This model is too complex with too many parameters and the trend does not reflect the reality of the data. So at k=1, the model is overfitting. When k gets larger, the deviation from true values starts to appear but the prediction still follows the general trend of the true values. When k = 50, the prediction tends to approach the average of a larger pool of data within the training dataset where a lot of irrelevant data are used. Thus, more errors are incurred with large k and the model is underfitting. The best k should be within the intermediate range.





**4. Making Predictions for Classifications**

**4.1 Prediction**

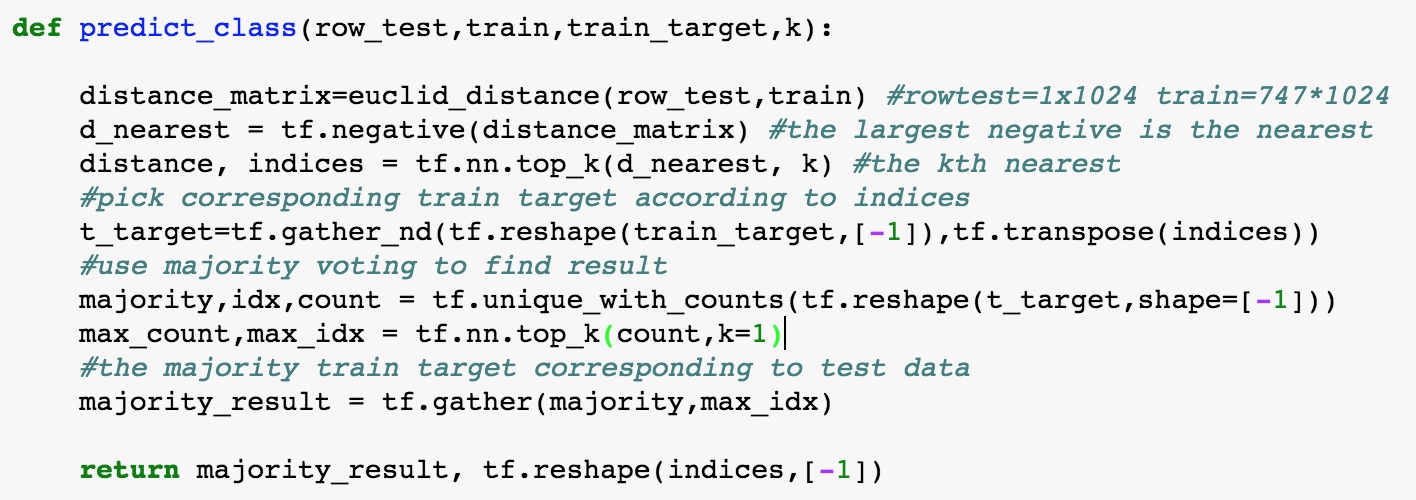
In order to make predictions for name and gender, the data are splitted into three portions, train data, valid data and test data. We use train data to train the model first, and then find the nearest k train data and corresponding train target to valid data. Lastly, majority voting is used to find the most frequent result in these k train target (Figure 4.1).

Figure 4.1 Prediction Function

**4.2 Find Accuracy**

K is chosen to be 1, 5, 10, 25, 50,100 and 200 . Afterwards, predicted results for valid data are compared with valid targets to determine the best k, and then with that k, test data is used to get test accuracy. To manipulate the test data easily, we chose to pass single test data (one image) to our predict function each time and iterate all test images outside the prediction function (Figure 4.2).





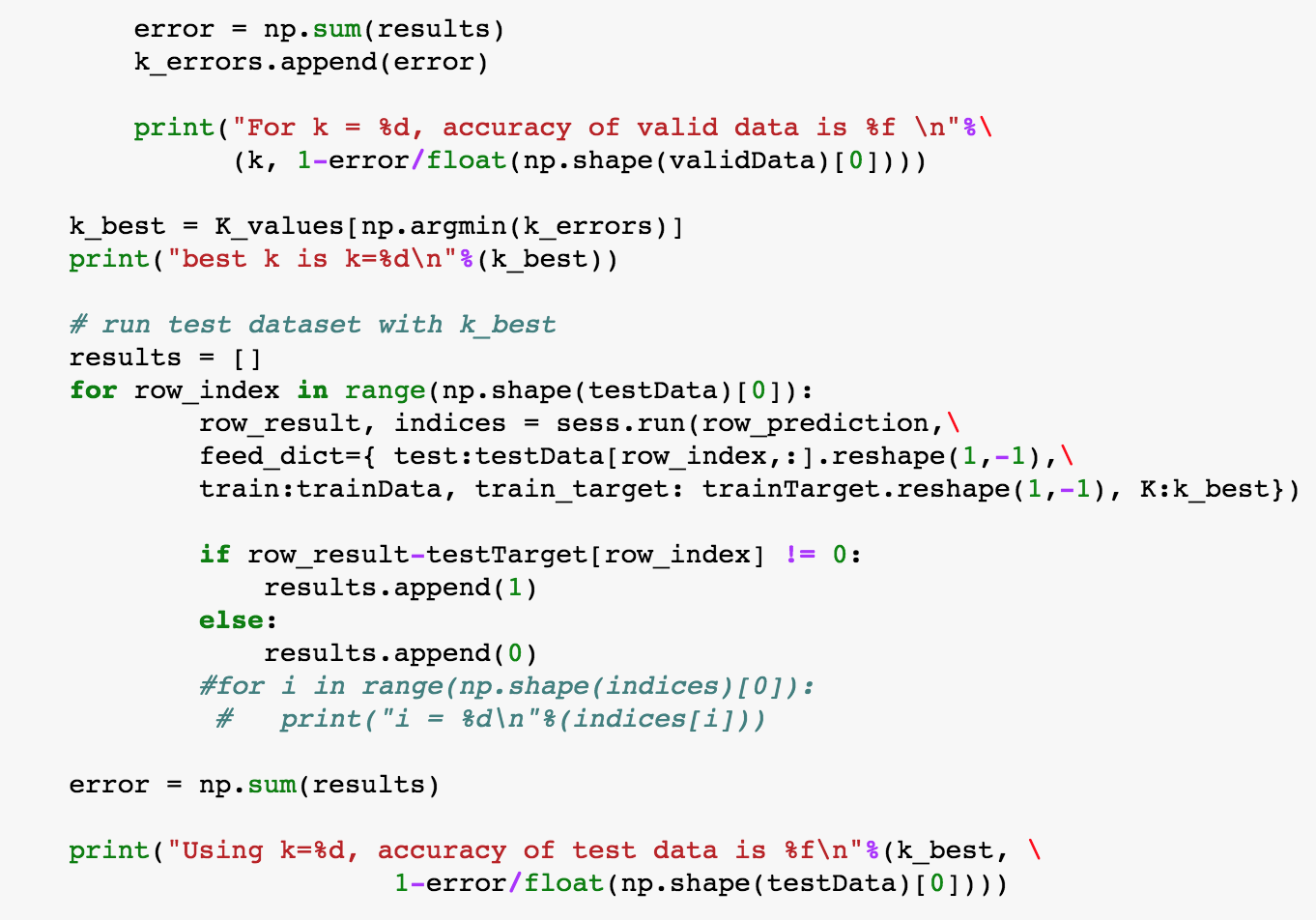


Figure 4.2 Load Data and find corresponding accuracy

**4.3 Conclusion**

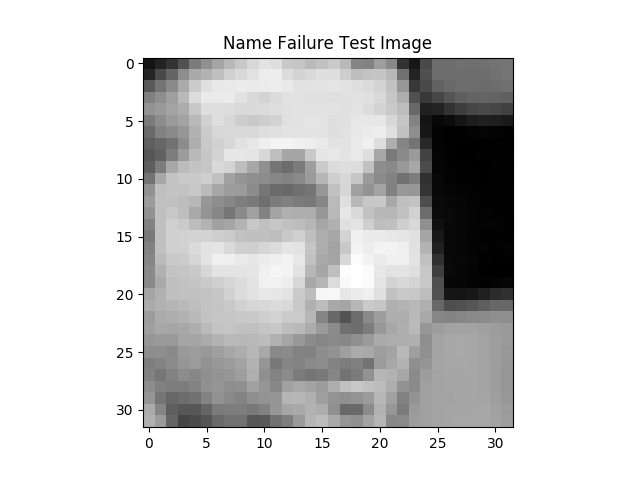
As shown in the table below, it is clear that k=1 is the best choice for both name prediction and gender prediction. With k=1, the accuracy of test data for name is 0.709677 and for gender is 0.924731. Generally speaking, the accuracy of predicted results for gender is much more higher than name. Because there are more classes for name test. And the accuracy decreased as increasing k, except for k=25, which has better accuracy than k=10 for both name and gender predictions.

|  |  |  |
| --- | --- | --- |
| **Face Recognition Accuracy** (Validation Data) | | |
|  | Name Recognition | Gender Recognition |
| k=1 | **0.663043** | **0.913043** |
| k=5 | 0.608696 | 0.913043 |
| k=10 | 0.576087 | 0.891304 |
| k=25 | 0.597826 | 0.902174 |
| k=50 | 0.576087 | 0.891304 |
| k=100 | 0.478261 | 0.858696 |
| k=200 | 0.315217 | 0.782609 |
| **Face Recognition Accuracy** (Test Data)  Using k=1 since it provides the highest accuracy | | |
| **k=1** | 0.709677 | 0.924731 |

Our algorithm picks the first incorrect prediction for both name and gender and it turns out to be same test data. As we can see in the 10 images below, most are female but the test image is male. In this case, the majority voting is wrong.

In conclusion, as the number of our train data are not large, k=1 has the best performance for both name and gender predictions.

Test image for both name and gender test:



Ten nearest train images for both name and gender test:

