Classifying Batting Success: Expectations vs. Reality

Data Study, MATH 498

Cooper Roach

April 29, 2025

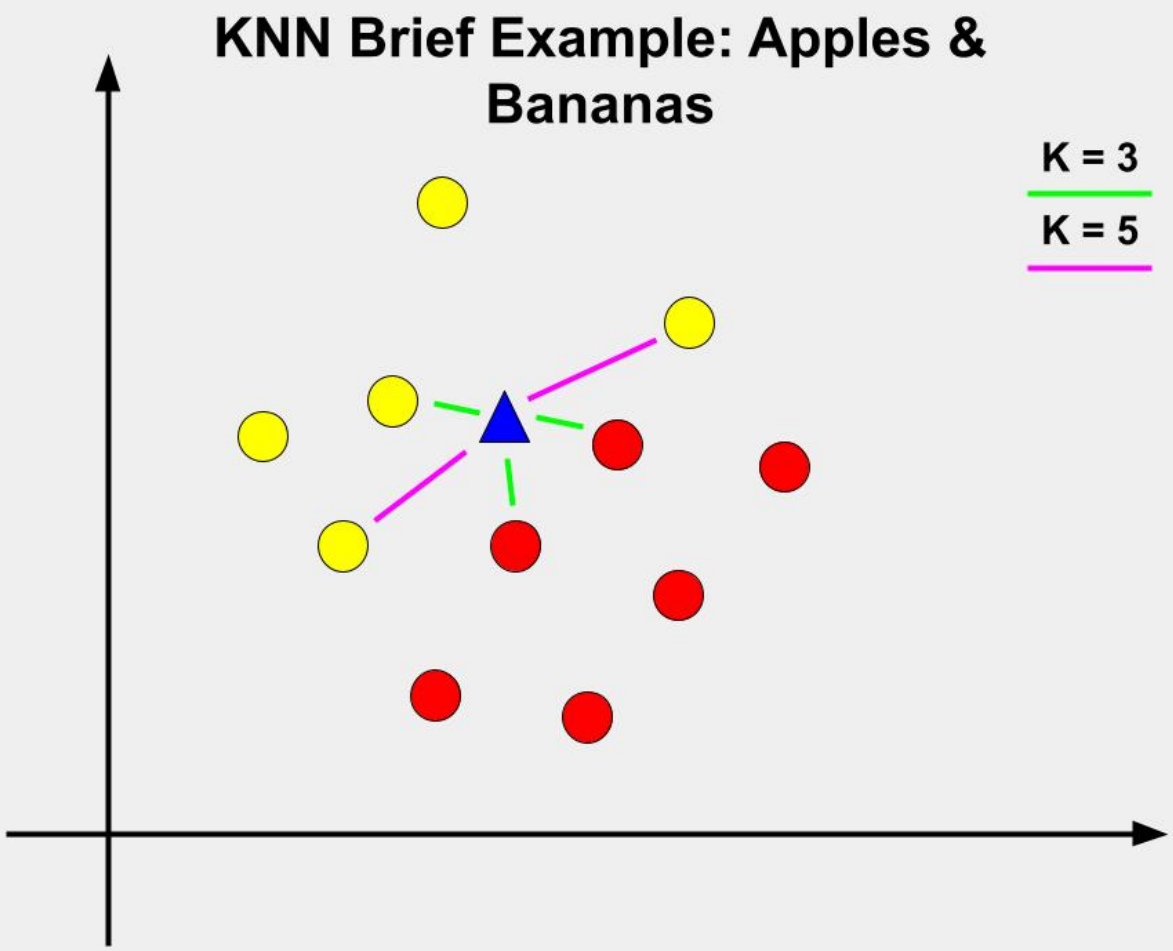
Table of Contents

Background

Background

- What is KNN?
  - Apple, Banana Example
  - Distance Formula
  - Normalization

Example Figure



How and Why?

- First, we will start with the 'how?'
- Second, we will learn about the 'why?'

How?

- This algorithm could use Mahalanobis distance which follows the formula:

$$D_M(x) = \sqrt{(x-\mu)^T \cdot S^{-1} \cdot (x-\mu)} \tag{1}$$

- x is the data point
- μ is the mean
- S^{-1} is the inverse of the covariance matrix
- $(x-\mu)^T$ is the transpose of the difference between the data point and the mean

How?

Instead, we are using simpler computations seen below:

- Manhattan distance:

$$D = \sum_{i=1}^{n} |x_i - y_i| \tag{2}$$

- Z-Score normalization:

$$X' = \frac{X - \mu}{\sigma} \tag{3}$$

Why?

- Mahalanobis distance:
  - Handles correlations and non-independent factors
  - Avoids the curse of dimensionality by using the covariance matrix
  - Slower computation
- Manhattan distance:
  - Computes the absolute distance between data points
  - Suffers from the curse of dimensionality (CoD) [1]
  - Fast computation
- Z-Score normalization:
  - Normalizes data so information can be measured easily

Dimensionality Reduction

- Principle component analysis (PCA)
  - Identification of variables that maintain and maximize variability while also being generally unrelated [2]

Results and Code

Z-Score Normalization Code

```
# Z normalize list of desired columns
def normalize(data, col_list):
    new_data = {}
    for column in data.columns:
        if column in col_list:
            col_data = []
            for index_r in range(data.shape[0]):
                z_normalize = ((data.loc[index_r, column] - data[column].mean())
                    / data[column].std())
                col_data.append(z_normalize)
                col_name = f'Znorm_{column}'
            new_data[col_name] = col_data
    return(new_data)
```

Figure: Z-Score Normalization

PCA: Volumetric vs. Quality Stats

```
pc1_sorted = loadings['PC1'].sort_values()
print(pc1_sorted[-40:])
```

```
barrel_batted_rate         0.074856
exit_velocity_avg          0.079534
xiso                       0.085246
popups                     0.087449
xwoba                      0.089961
slg_percent                0.090147
woba                       0.090236
b_total_swinging_strike    0.090818
xslg                       0.090950
on_base_plus_slg           0.091550
groundballs                0.108023
single                     0.126183
b_foul_tip                 0.130517
in_zone_swing_miss         0.134526
b_swinging_strike          0.137320
b_called_strike            0.138444
pitch_count_offspeed       0.142598
out_zone_swing             0.143485
```

Figure: PCA Result

```
out_zone_swing             0.143485
strikeout                  0.147364
linedrives                 0.148196
walk                       0.154411
home_run                   0.157949
barrel                     0.158840
double                     0.158951
batted_ball                0.162860
flyballs                   0.168199
hit                        0.180444
b_foul                     0.182095
pitch_count_breaking       0.182820
pitch_count_fastball       0.187990
ab                         0.188835
in_zone                    0.192059
r_run                      0.192276
in_zone_swing              0.195078
b_ball                     0.195669
b_total_strike             0.197269
b_total_ball               0.197814
pa                         0.198972
pitch_count                0.203916
out_zone                   0.206485
```

Figure: PCA Result Continued

KNN: Manhattan Distance Code 1

```python
def manhattan_distance(data, player1_id, player1_year, player2):
    player1 = data.loc[get_player_stats(data, player1_id, player1_year)].iloc[0]
    woba = abs(player1['Znorm_woba'] - data.loc[player2, 'Znorm_woba'])
    ops = abs(player1['Znorm_on_base_plus_slg'] - data.loc[player2, 'Znorm_on_base_plus_slg'])
    hhp = abs(player1['Znorm_hard_hit_percent'] - data.loc[player2, 'Znorm_hard_hit_percent'])
    return (woba + ops + hhp)
```

Figure: Manhattan Distance Code

KNN: Manhattan Distance Code 2

```python
def knn_predict(data, k, player_id, player_year):
    distances = []
    for i in range(data.shape[0]):
        dist = manhattan_distance(data, 701538, 2024, i)
        success_level = data.iloc[i]['success']
        distances.append([dist, success_level])

    dist_sorted = sorted(distances, key=lambda x: x[0], reverse = True)
    k_nearest_labels = [label for _, label in dist_sorted[:k]]
    print(k_nearest_labels)
    prediction = max(set(k_nearest_labels), key=k_nearest_labels.count)
    return prediction
```

Figure: KNN Prediction Code

Jackson Merrill Results

```python
# knn for Jackson Merrill, 2024 using Manhattan
# Expected Success Level: 4
prediction = knn_predict(knn_data, k, 701538, 2024)
print(f'Success level: {prediction}')
```

```
[4, 1, 4, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1]
Success level: 1
```

```python
# knn for Jackson Merrill, 2024 using Manhattan
# Expected Success Level: 4
prediction = knn_predict(knn_data, 5, 701538, 2024)
print(f'Success level: {prediction}')
```

```
[4, 1, 4, 4, 1]
Success level: 4
```

Figure: Merrill Predictions

Robbie Grossman Results

```python
# knn for Robbie Grossman, 2021 using Manhattan
# Expected Success Level: 2
prediction = knn_predict(knn_data, k, 543257, 2021)
print(f'Success level: {prediction}')
```

```
[4, 1, 4, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1]
Success level: 1
```

```python
# knn for Robbie Grossman, 2021 using Manhattan
# Expected Success Level: 2
prediction = knn_predict(knn_data, 11, 543257, 2021)
print(f'Success level: {prediction}')
```

```
[4, 1, 4, 4, 1, 1, 1, 1, 1, 1, 1]
Success level: 1
```

Figure: Grossman Predictions

KNN: Mahalanobis Distance Code 1

```python
def mahalanobis_distance(data, player1_id, player1_year, player2, inv_cov_matrix):
    player1_stats = data.loc[get_player_stats(data, player1_id, player1_year)].iloc[0]
    player2_stats = data.loc[player2]
    diff = np.array([
        player1_stats['Znorm_woba'] - player2_stats['Znorm_woba'],
        player1_stats['Znorm_on_base_plus_slg'] - player2_stats['Znorm_on_base_plus_slg'],
        player1_stats['Znorm_hard_hit_percent'] - player2_stats['Znorm_hard_hit_percent']
    ])
    mahalanobis_dist = np.sqrt(np.dot(np.dot(diff.T, inv_cov_matrix), diff))
    return mahalanobis_dist
```

Figure: Mahalanobis Code

KNN: Mahalanobis Distance Code 2

```python
def knn_predict2(data, k, player_id, player_year, inv_cov_matrix):
    distances = []
    for i in range(data.shape[0]):
        dist = mahalanobis_distance(data, player_id, player_year, i, inv_cov_matrix)
        success_level = data.iloc[i]['success']
        distances.append([dist, success_level])

    dist_sorted = sorted(distances, key=lambda x: x[0], reverse = True)
    k_nearest_labels = [label for _, label in dist_sorted[:k]]
    print(k_nearest_labels)
    prediction = max(set(k_nearest_labels), key=k_nearest_labels.count)
    return prediction
```

Figure: KNN Prediction Code

Jackson Merrill Results

```
# knn for Jackson Merrill, 2024 using Mahalanobis
# Expected Success Level: 4
prediction = knn_predict2(knn_data, k, 701538, 2024, inverse_covariance_matrix)
print(f'Success level: {prediction}')
```

```
[1, 4, 1, 2, 4, 1, 1, 4, 4, 2, 1, 1, 1, 1, 1, 2, 4, 4, 1, 4, 1, 2, 2, 1, 1, 1,
1, 4, 1, 4, 1, 1, 2, 1, 1, 1, 2]
Success level: 1
```

```
# knn for Jackson Merrill, 2024 using Mahalanobis
# Expected Success Level: 4
prediction = knn_predict2(knn_data, 7, 701538, 2024, inverse_covariance_matrix)
print(f'Success level: {prediction}')
```

```
[1, 4, 1, 2, 4, 1, 1]
Success level: 1
```

Figure: Merrill Prediction

Robbie Grossman Results

```python
# knn for Robbie Grossman, 2021 using Mahalanobis
# Expected Success Level: 2
prediction = knn_predict2(knn_data, k, 543257, 2021, inverse_covariance_matrix)
print(f'Success level: {prediction}')
```

```
[4, 4, 4, 4, 1, 4, 4, 4, 3, 1, 4, 4, 1, 1, 3, 2, 4, 4, 1, 1, 1, 1, 3, 2, 4, 4,
4, 2, 4, 3, 3, 4, 3, 2, 4, 4, 4]
Success level: 4
```

```python
# knn for Robbie Grossman, 2021 using Mahalanobis
# Expected Success Level: 2
prediction = knn_predict2(knn_data, 11, 543257, 2021, inverse_covariance_matrix
print(f'Success level: {prediction}')
```

```
[4, 4, 4, 4, 1, 4, 4, 4, 3, 1, 4]
Success level: 4
```

Figure: Grossman Prediction

Discussion

Discussion

Limitations of this study

- Curse of dimensionality
  - Washed out distances
- Hardware restrictions
- Access to information
  - Limited information
  - Data accuracy

Conclusions

Conclusions

- KNN was not a good fit for this section of statistics
- Alternative Algorithms
  - Random forest
  - Shallow neural network

Links and References

Links

- Data Source: Baseball Savant
- Project Github Repository: CRoach02 / Math 498

References

[1] N. Altman and M. Krzywinski, **"The curse(s) of dimensionality,"** *Nature Methods*, vol. 15, pp. 397–400, 2018.

[2] I. T. Jollife and J. Codima, **"Principle component analysis: A review and recent developments,"** *Philosophical transactions of the Royal Society of London. Series A: Mathematical, physical, and engineering sciences*, vol. 374, p. 2 015 202, 2016.