148 HW 2
Margaret Capetz

1. RMSE for KNN

Root mean square error

$$\sqrt{\frac{1}{6}\left((1-1)^2+(4-4)^2+(6-6)^2+(3-3)^2+(2-2)^2+(2-2)^2\right)}$$

$= 0$

$$\sqrt{\frac{1}{6}\left((1-2.5)^2+(4-4)^2+(6-4.5)^2+(3-3)^2+(2-2)^2+(2-2)^2\right)}$$

$= 0.866025403784$

$$\sqrt{\frac{1}{6}\left(\left(1-\frac{11}{3}\right)^2+\left(4-\frac{11}{3}\right)^2+\left(6-\frac{13}{3}\right)^2+\left(3-\frac{11}{3}\right)^2+\left(2-\frac{7}{3}\right)^2+\left(2-\frac{7}{3}\right)^2\right)}$$

$= 1.33333333333$

$$\sqrt{\frac{1}{6}\left((1-3)^2+(4-3)^2+(6-3)^2+(3-3)^2+(2-3)^2+(2-3)^2\right)}$$

$= 1.63299316186$

Training data:
K = 1: RSME = 0
K = 2: RSME = 0.866
K = 3: RSME = 1 ⅓
K = 6: RSME: 1.633

K = 1 minimizes the RSME for the training data. However, K = 1 overfits the data so if I were to choose beyond just looking at the minimum RMSE, I would choose K = 2.

$$\sqrt{\frac{1}{3}\left((2-1)^2+(5-6)^2+(2.5-3)^2\right)}$$

$= 0.866025403784$

$$\sqrt{\frac{1}{3}\left((2-2.5)^2+(5-4.5)^2+(2.5-2.5)^2\right)}$$

$= 0.408248290464$

$$\sqrt{\frac{1}{3}\left(\left(2-\frac{11}{3}\right)^2+\left(5-\frac{13}{3}\right)^2+\left(2.5-\frac{11}{3}\right)^2\right)}$$

$= 1.23603308118$

$$\sqrt{\frac{1}{3}\left((2-3)^2+(5-3)^2+(2.5-3)^2\right)}$$

$= 1.32287565553$

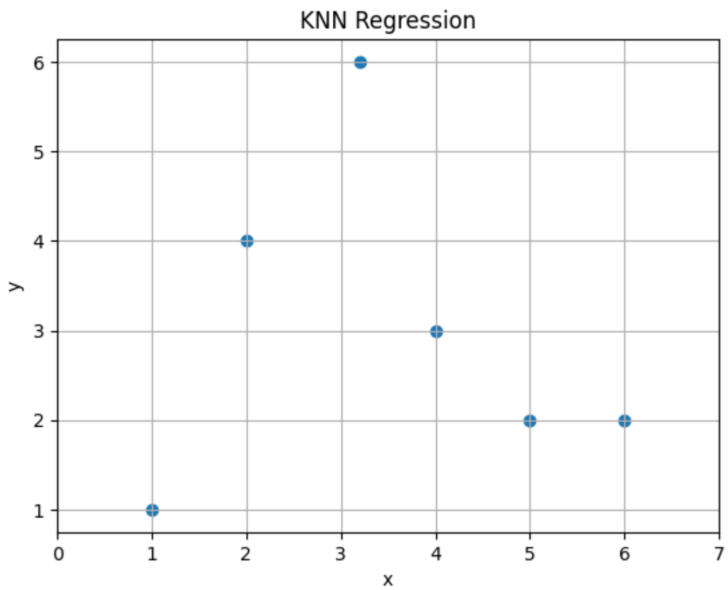
Testing data: A = {(x, y)} = {(1.25, 2),(3.4, 5),(4.25, 2.5)}.
K = 1: RSME = 0.866
K = 2: RSME = 0.408
K = 3: RSME = 1.236
K = 6: RSME: 1.323

K = 2 minimizes the RSME for the testing data.
See end of PDF for code.

Question 1: Root Mean Squared Error for KNN

```
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
import numpy as np
```

```
x = [[1], [2], [3.2], [4], [5], [6]]
y = [1, 4, 6, 3, 2, 2]
plt.scatter(x, y)
plt.grid()
plt.title("KNN Regression")
plt.xlabel("x")
plt.ylabel("y")
plt.xlim([0, 7])
plt.show()
```



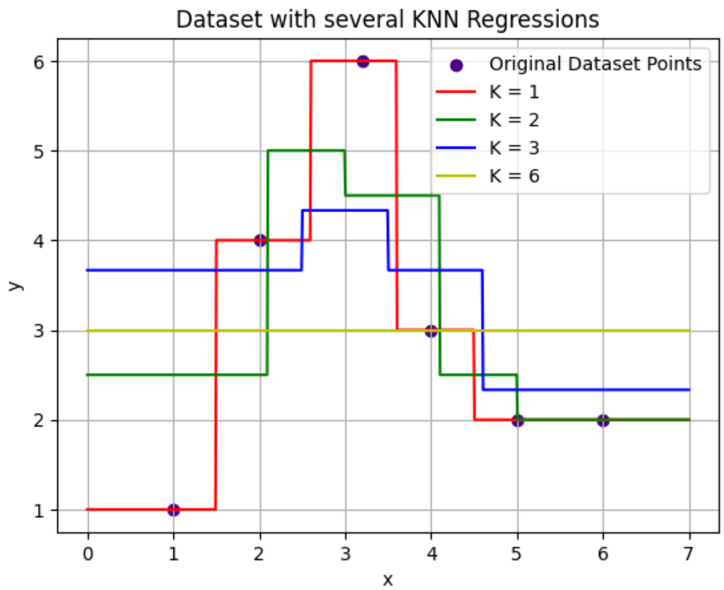

```
k1 = KNeighborsRegressor(n_neighbors = 1)
k1.fit(x, y)
k2 = KNeighborsRegressor(n_neighbors = 2)
k2.fit(x, y)
k3 = KNeighborsRegressor(n_neighbors = 3)
k3.fit(x, y)
k6 = KNeighborsRegressor(n_neighbors =6)
k6.fit(x, y)
```

```
▾        KNeighborsRegressor
KNeighborsRegressor(n_neighbors=6)
```

```
pred_y = np.linspace(0, 7, 700)
k1list = []
k2list = []
k3list = []
k6list = []

for i in range (len(pred_y)):
  k1list.append(k1.predict([[pred_y[i]]]))
  k2list.append(k2.predict([[pred_y[i]]]))
  k3list.append(k3.predict([[pred_y[i]]]))
  k6list.append(k6.predict([[pred_y[i]]]))
```

```
plt.title("Dataset with several KNN Regressions")
plt.xlabel("x")
plt.ylabel("y")
plt.scatter(x, y, color = "indigo", label = "Original Dataset Points")
plt.plot(pred_y, k1list, label = "K = 1", color = "r")
plt.plot(pred_y, k2list, label = "K = 2", color = "g")
plt.plot(pred_y, k3list, label = "K = 3", color = "b")
plt.plot(pred_y, k6list, label = "K = 6", color = "y")
plt.legend()
```

```
plt.grid()
plt.show()
```




```
def rmse(x, y, largex, knn):
  sum = 0
  length = len(y)
  current = 0
  for i in range (len(largex)):
    if(current >= length):
      return np.sqrt(sum/length)
    if round(largex[i],2) == x[current]:
      sum += (y[current]-knn[i])**2
      current += 1

  return np.sqrt(sum/length)


x_original = [1, 2, 3.2, 4, 5, 6]
y_original = [1, 4, 6, 3, 2, 2]

x_new = [1.25, 3.4, 4.25]
y_new = [2, 5, 2.5]


arr = [k1list, k2list, k3list, k6list]

print("original data")
for k in arr:
  print(rmse(x_original, y_original, pred_y, k))

print("new data")
for k in arr:
  print(rmse(x_new, y_new, pred_y, k))
```

```
original data
[0.]
[1.2416387]
[1.33333333]
[1.63299316]
new data
[0.8660254]
[0.40824829]
[1.23603308]
[1.32287566]
```

Question 2. Mean Square Error

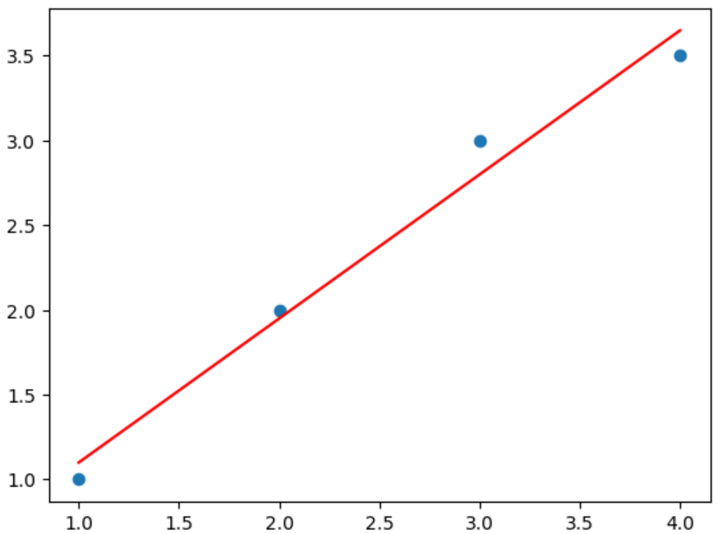
```
x = [1, 2, 3, 4]
y = [1, 2, 3, 3.5]


def b1(x, y):
  xavg = sum(x) / len(x)
  yavg = sum(y) / len(y)
  length = len(x)
```

```
  numerator = 0.0
  denominator = 0.0
  for i in range(length):
    numerator += (x[i]-xavg)*(y[i]-yavg)
    denominator += (x[i]-xavg)**2

  return numerator/denominator

def b0(x,y):
  xavg = sum(x) / len(x)
  yavg = sum(y) / len(y)
  return yavg - b1(x,y)*xavg
```

```
print(b1(x,y), b0(x,y))
```

```
0.85 0.25
```

```
largex = np.linspace(1, 4, 300)
largey = largex * b1(x, y) + b0(x, y)
plt.scatter(x, y)
plt.plot(largex, largey, color = "red")
plt.show()
```




```
def r2(x, y, largex, largey):
  numerator = 0.0
  denominator = 0.0
  xavg = np.average(x)
  yavg = np.average(y)
  curr = 0

  for i in range(len(y)):
    denominator += (yavg - y[i])**2

  for j in range(len(largex)):
    if curr >= len(x):
      return 1-(numerator/denominator)
    if largex[j] == x[curr]:
      numerator += (largey[j]-y[curr])**2
      curr += 1

  return 1-(numerator/denominator)
```

```
print(r2(x,y,largex,largey))
```

```
0.9972881355932204
```

0s completed at 10:02 AM

2. MSE

2. Suppose we have the following data points with coordinates $(x, y) : \{(1,1),(2,2),(3,3),(4,3.5)\}$.

   (a) Suppose you want to fit the model $Y = \beta_0 + \beta_1 \times X$ by minimizing the mean square error (MSE) $\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 \times x_i)^2$. Write down the conditions for the derivative of the MSE that is necessary for β_0, β_1 to be optimal. From the conditions on the derivative, derive the formulae for β_0 and β_1. You do not need to re-derive the exact equations shown in class but you must derive a closed form solution for β_0 and β_1 in terms of the data (x, y).

   Hint: The following equalities may prove useful $\sum_{i=1}^{n}(\overline{x})^2 - \overline{x}x_i = 0$ and $\sum_{i=1}^{n}\overline{y}\,\overline{x} - y_i\overline{x} = 0$ where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

   (b) Fit the model $Y = \beta_0 + \beta_1 \times X$ based on the given data points by minimizing MSE. Compute R^2 for this model and briefly explain the meaning of the parameter β_1.

Derivation

$$\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)^2 = 0$$

$$= \frac{\partial}{\partial\beta_0}\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)^2$$

$$= -2\left(\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)\right)$$

$$= -2(\bar{y}-\beta_0-\beta_1\bar{x}) \Rightarrow \boxed{\beta_0 = \bar{y}-\beta_1\bar{x}}$$

$$= \frac{\partial}{\partial\beta_1}\left(-2\left(\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)\right)\right)$$

$$= -2\left(\frac{1}{n}\sum_{i=1}^{n}x_i(y_i-\beta_0-\beta_1x_i)\right)$$

$$= -2\left(\frac{1}{n}\sum_{i=1}^{n}x_iy_i-\beta_0\bar{x}-\beta_1\sum_{i=1}^{n}\bar{x}^2\right.$$

$$= -2\left(\frac{1}{n}\sum_{i=1}^{n}x_iy_i-\bar{x}(\bar{y}-\beta_1\bar{x})-\beta_1\left(\frac{1}{n}\sum_{i=1}^{n}x_i^2\right)\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n}x_iy_i-\bar{x}\bar{y}+\beta_1\bar{x}^2-\beta_1\left(\frac{1}{n}\sum_{i=1}^{n}x_i^2\right)$$

$$\beta_1\left(\bar{x}^2-\frac{1}{n}\sum_{i=1}^{n}x_i^2\right) = \bar{x}\bar{y}-\frac{1}{n}\sum_{i=1}^{n}x_iy_i$$

$$\beta_1 = \frac{\bar{x}\bar{y}-\frac{1}{n}\sum_{i=1}^{n}x_iy_i}{\bar{x}^2-\frac{1}{n}\sum_{i=1}^{n}x_i^2}$$

$$= \frac{-n\bar{x}\bar{y}+\sum_{i=1}^{n}x_iy_i}{-n\bar{x}^2+\sum_{i=1}^{n}x_i^2}$$

$$= \frac{-n\bar{x}\bar{y}-n\bar{x}\bar{y}+n\bar{x}\bar{y}+\sum_{i=1}^{n}x_iy_i}{-2n\bar{x}^2+n\bar{x}^2+\sum_{i=1}^{n}x_i^2}$$

$$= \frac{\sum_{i=1}^{n}x_iy_i-\bar{y}\sum_{i=1}^{n}x_i-\bar{x}\sum_{i=1}^{n}y_i+n\bar{x}\bar{y}}{-2n\frac{1}{n}\sum_{i=1}^{n}x_i\cdot\bar{x}+\bar{x}^2+\sum_{i=1}^{n}x_i^2}$$

$$\boxed{\beta_1 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}$$

$\frac{1}{4}(1+2+3+3.5)$

$= 2.375$

$\frac{(1-2.5)\cdot(1-2.375)+(2-2.5)\cdot(2-2.375)+(3-2.5)\cdot(3-2.375)+(4-2.5)\cdot(3.5-2.375)}{(1-2.5)^2+(2-2.5)^2+(3-2.5)^2+(4-2.5)^2}$

$= 0.85$

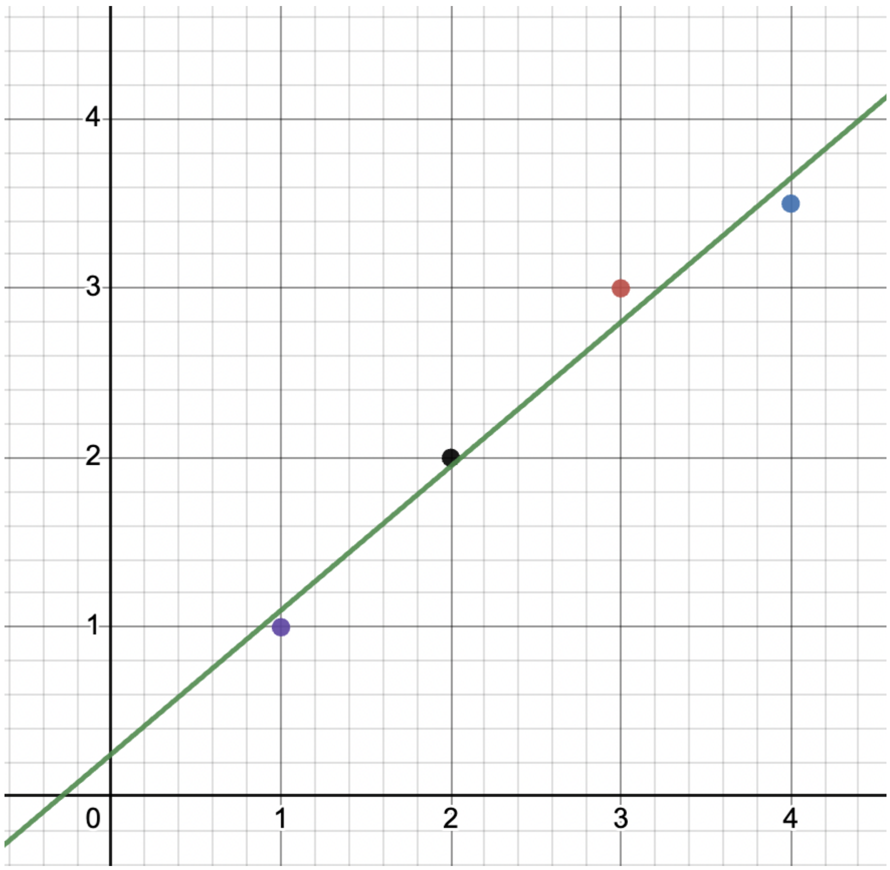
$2.375 - 0.85\cdot 2.5$

$= 0.25$

X_avg = 2.5

Y_avg = 2.375

Beta_1 = 0.85
Beta_0 = 0.25



$$1 - \frac{(1.1-1)^2 + (1.95-2)^2 + (2.8-3)^2 + (3.65-3.5)^2}{(2.375-1)^2 + (2.375-2)^2 + (2.375-3)^2 + (2.375-3.5)^2}$$

$$= 0.979661016949$$

R^2 = 0.98, which is very close to 1, indicating the data is very positively correlated.

The beta_1 parameter measures the slope of the line of best fit (linear model) for the data. In other words, beta_1 measures how much one feature (x-axis) dictates the outcome of another (y-axis), aka its “weight.”

3. One-hot encoding
    a. One-hot encoding is the strategy of using vectors with binary entries (1 or 0) to represent categorical variables, in which there is a single entry of 1 and the rest of the entries are 0. The position of the 1 entry indicates a certain feature.
    b. Is one-hot encoding appropriate for the following?
        i. Zipcode of the house: Yes, if zipcode is treated as a categorical variable.
        ii. Price of the house: No, since price is a continuous variable.
        iii. City of the house: Yes, with each city being a binary vector. Each house can only belong to one city (by assumption) thus one hot encoding is suitable.
        iv. Name of homeowner (assume each homeowner owns only one home): No, one hot encoding is only recommended when the number of categories is much smaller than the number of data points/ observations (rows of the dataset). Thus one hot encoding is not suitable for the name of the homeowner.
        v. Year the house was built: No, since price is a continuous variable.

4. Fitting
    a. Overfit, since the line perfectly fits through each data point. The line is not generalized.
    b. Good fit, it doesn’t exactly fit the data and provides good generality.
    c. Underfit, the plot is too generalized and doesn’t accurately represent the data.

5. True or false
    a. T, we can achieve the best linear model by selecting parameters that minimize the loss function.
    b. F, in the case of overfitting the training data error is low and the testing data error is high.
    c. T, R^2 measures the correlation of the data and has the domain [0,1].
    d. F, multilinear regression is multi dimensional linear regression. In this case, polynomial regression is a special case of multilinear regression.
    e. F, as K gets larger, the regression model becomes more underfit to the data.