การบ้านที่ 3 (1/2564)

ข้อที่ 2

2.1 ให้ออกแบบและเขียนโปรแกรมภาษา Python สำหรับสร้างโมเดลเพื่อแก้ปัญหาในข้อ 1.A 1.B. และ 1.C โดยสามารถเลือกใช้ Classifier ที่ได้เรียนมา ตามที่นักศึกษาเห็นว่าเหมาะสมกับปัญหา (Naïve Bayes, Linear Regression, Logistic Regression) แต่จะต้องระบุปัญหาและ Classifier ที่เลือกอย่างชัดเจน

ปัญหาข้อ 1.A และ 1.C จะใช้ Naïve Bayes ในการแก้ไขปัญหา

ปัญหาข้อ 1.B จะใช้ Linear Regression ในการแก้ไขปัญหา

2.2 จากข้อ 2.1 ให้แยกเป็น training และ test sets ในสัดส่วนที่เหมาะสม เช่น 70:30, 80:20 เป็นต้น โดยให้ใช้ training set ในการ train โมเดล และ ใช้ test set ในการ evaluate ประสิทธิภาพของโมเดลที่เลือก โดยให้รายงานผลประสิทธิภาพของโมเดลเป็นค่าความแม่นยำ (Accuracy) สำหรับปัญหาที่เป็น Classification และค่า Root Mean Square Error (RMSE) สำหรับปัญหาที่เป็น Regression

ปัญหาข้อ 1.A หลังจากการแยกเป็น training และ test sets ในสัดส่วน 80:20 และนำ training set ไป train Model Naïve Bayes และใช้ test set ในการแก้ไขปัญหา พบว่า Model Naïve Bayes มีประสิทธิภาพความแม่นยำ (Accuracy) อยู่ที่ 0.9666666666666667

ปัญหาข้อ 1.B หลังจากการแยกเป็น training และ test sets ในสัดส่วน 80:20 และนำ training set ไป train Model Linear Regression และใช้ test set ในการแก้ไขปัญหา พบว่า Model Linear Regression มีค่า Root Mean Square Error (RMSE) อยู่ที่ 0.7115124735378854

ปัญหาข้อ 1.C หลังจากการแยกเป็น training และ test sets ในสัดส่วน 80:20 และนำ training set ไป train Model Naïve Bayes และใช้ test set ในการแก้ไขปัญหา พบว่า Model Naïve Bayes มีประสิทธิภาพความแม่นยำ (Accuracy) อยู่ที่ 0.7291666666666666

2.3 ให้ระบุรายละเอียดของ Setting ต่างๆ ของโมเดลที่เขียน รวมถึง อภิปรายผลลัพธ์ที่ได้อย่างละเอียด

Model Naïve Bayes ได้ใช้วิธีคิดจาก Naïve Bayes Algorithm โดยหาผลลัพธ์จากการหาด้วย argmax

$$Y = argmax_{y_k} P(Y = y_k) \prod_i P(X_i | Y = y_k)$$

ซึ่งใช้ Model Naïve Bayes ในการแก้ปัญหาในข้อ 1.A และ 1.C ซึ่งข้อมูลทั้ง 2 มีข้อมูลทั้งแบบ discrete และ continuous โดยแบบ discrete ใช้วิธีการหาค่าแบบ Maximum likelihood estimates (MLE)

$$\hat{\theta} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D\{X_i = x_{ij} \text{ and } Y = y_k\}}{\#D\{Y = y_k\}}$$

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$$

แต่ไม่ได้มีการจัดการในเรื่องของหากค่าประมาณของพารามิเตอร์เป็น 0 จะต้องบวกค่าเพิ่มเติม

และแบบ continuous ได้ใช้วิธีการหาค่าจาก Gaussian Distribution โดยต้องหาค่า mean และ S.D. ก่อน

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

$$\mu_{ik} = E[X_i | Y = y_k]$$
$$\sigma_{ik}^2 = E[((X_i - \mu_{ik})^2 | Y = y_k)]$$

โดย Model Naïve Bayes ที่สร้างนั้น ได้สร้างเป็น class โดยมี 2 ฟังก์ชัน ได้แก่

1. fit(X, Y) จะเป็นฟังก์ชันในการ fit ข้อมูลที่ส่งเข้าไปกับ model โดย model จะยึดข้อมูลที่ส่งเข้าไป ไปทำการ train กับ model ที่มีอยู่

2. predict(test) จะเป็นฟังก์ชันในการ predict ข้อมูล โดย predict ด้วยการดูจากข้อมูลที่ส่งเข้าไปคู่กับข้อมูลที่ได้มีการ train ไว้แล้ว หลังจาก predict เสร็จ จะทำการส่งผลลัพธ์คืนไป

ทำการส่งชุดข้อมูลจากข้อ 1.A ด้วยการ fit จากนั้นทำการ predict เมื่อได้ผลลัพธ์แล้ว ก็ทำการวัดประสิทธิภาพของ model ด้วยฟังก์ชัน accuracy_score จาก library sklearn ซึ่งได้ค่า accuracy เท่ากับ 0.9666666666666667 มีค่าเข้าใกล้ 1 มาก แปลว่าค่าที่ทำนายมา มีผลลัพธ์เกือบจะเหมือนกับค่าจริงเลย หรือก็คือ จากชุดข้อมูล 1.A Model Naïve Bayes มีประสิทธิภาพดีเยี่ยม

และทำการส่งชุดข้อมูลจากข้อ 1.C ด้วยการ fit จากนั้นทำการ predict เมื่อได้ผลลัพธ์แล้ว ก็ทำการวัดประสิทธิภาพของ model ด้วยฟังก์ชัน accuracy_score จาก library sklearn ซึ่งได้ค่า accuracy เท่ากับ 0.7291666666666666 ผลลัพธ์ที่ทำนายออกมา ทำนายถูกถึง 7 ใน 10 ของข้อมูล ก็อาจกล่าวได้ว่า จากชุดข้อมูล 1.C Model Naïve Bayes มีประสิทธิภาพดี

ต่อมา Model Linear Regression โดยข้อมูลของ 1.B จะมีหลาย feature ทำให้เลือกใช้เป็น Multiple Linear Regression โดยได้ใช้สมการดังภาพนี้

$$h(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x$$
$$\text{Where } x_0 = 1$$

โดยการทำ Linear Regression จะต้องหาค่า θ ที่ทำให้ J(θ) มีค่าน้อยที่สุด โดยสมการของ J(θ) จะใช้ดังภาพนี้

$$J(\theta) = \frac{1}{2} \sum_{i=0}^{m} (h_\theta(x) - y)^2)$$

วิธีการหา θ จะใช้วิธีที่ชื่อ Gradient Descent Algorithm ซึ่งวิธีการดังกล่าวจะมีวิธีให้เลือกหลายวิธี โดยที่เลือกใช้จะเป็น Batch Gradient Descent, Stochastic Gradient Descent และ Mini Batch Gradient Descent

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

Batch Gradient Descent

$$\theta_j := \theta_j - \alpha (h_\theta(x^i) - y^i) x_j^i$$

Stochastic Gradient Descent

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Mini Batch Gradient Descent

โดย Model Linear Regression ที่สร้างนั้น ได้สร้างเป็น class ที่สามารถกำหนดพารามิเตอร์ epochs, learning_rate, method และ batch_size ได้ โดยแต่ละพารามิเตอร์มีความหมายดังนี้

1. epochs จะเป็นจำนวนรอบในการวนทำหรืออาจเรียกเป็น iteration ได้ หากไม่ได้กำหนด จะมีกำหนดเป็น default ไว้ที่ 1,000
2. learning_rate จะเป็นหนึ่งในค่าที่เอาไว้เปลี่ยนค่า θ หากไม่ได้กำหนด จะมีกำหนดเป็น default ไว้ที่ -99 (เมื่อทำการ fit หากค่าอยู่ที่ -99 จะเปลี่ยนเป็น 1 ไว้ป้องกันสำหรับคนที่กำหนด learning_rate = 1 ซึ่งเป็นค่าที่ทาง model กำหนดไว้) ซึ่งค่า learning_rate หากไม่ได้กำหนดค่า ในขั้นตอนการ fit จะทำการวนหาค่า best learning rate หรือ ค่า learning rate ที่เหมาะสม โดยค่าจะเริ่มที่ 1 แล้วหากค่ายังไม่เหมาะสม จะทำการหาร 1.1 ไปเรื่อย ๆ จนกว่าจะเจอค่าที่เหมาะสม (ค่าที่เหมาะสมคือค่าที่เมื่อนำไปหา θ ตัวใหม่แล้วทำให้ค่า $J(\theta)$ ลดลง)

3. method จะเป็นการกำหนดวิธีของ Gradient Descent Algorithm 0 คือ Batch Gradient Descent 1 คือ Stochastic Gradient Descent และ 2 คือ Mini Batch Gradient Descent หากไม่ได้กำหนดจะมี default อยู่ที่ 0 นั้นคือ จะทำ Batch Gradient Descent เป็น default
4. batch_size คือขนาดของ batch ที่จะใช้ใน Mini Batch Gradient Descent หากไม่ได้กำหนดจะมี default อยู่ที่ 0

ซึ่ง Model จะมี 2 ฟังก์ชัน ได้แก่

1. fit(X, Y) จะเป็นฟังก์ชันในการ fit ข้อมูลที่ส่งเข้าไปกับ model โดย model จะยึดข้อมูลที่ส่งเข้าไป ไปทำการ train
2. predict(test) จะเป็นฟังก์ชันในการ predict ข้อมูล โดย predict ด้วยการดูจากข้อมูลที่ส่งเข้าไปคู่กับข้อมูลที่ได้มีการ train ไว้แล้ว หลังจาก predict เสร็จ จะทำการส่งผลลัพธ์คืนไป

ทำการส่งชุดข้อมูลจากข้อ 1.B ด้วยการ fit โดยเลือกใช้วิธี Batch Gradient Descent (ได้ผลลัพธ์ดีที่สุดจากการทดลองทั้ง 3 วิธี) จากนั้นทำการ predict เมื่อได้ผลลัพธ์แล้ว ก็ทำการหาค่า Root Mean Square Error (RMSE) โดยหาค่าจาก 2 วิธี

1. เขียนโค้ดโดยดูจากสมการ $\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$ ซึ่งได้ผลลัพธ์คือ 0.7115124735378854
2. ใช้ฟังก์ชัน mean_squared_error จาก library sklearn โดยหากกำหนดพารามิเตอร์ squared = False ผลลัพธ์จะได้ออกมาเป็นค่าของ RMSD (หากกำหนดเป็น True จะได้เป็น MSD) ซึ่งได้ค่า rmsd เท่ากับ 0.7115124735378854

ซึ่งค่าที่ได้ออกมานั้นหมายถึงค่าจริงกับค่าที่ทำนายออกมาแตกต่างกันขนาดไหน โดยได้ผลลัพธ์คือ 0.7115124735378854 ซึ่งค่าใกล้ 0 อยู่พอสมควร (ยิ่งใกล้ 0 แปลว่าค่าที่ทำนายออกมานั้นใกล้กับค่าจริงมาก ๆ) อาจกล่าวได้ว่า จากชุดข้อมูล 1.B Model Linear Regression มีประสิทธิภาพดี

2.4 ให้เลือก 1 ปัญหาจาก 1.A 1.B. และ 1.C ที่เป็น Regression

2.4.1 ให้ออกแบบการทดลองเพื่อเปรียบเทียบค่า RMSE เฉลี่ย ของ Training set และ Test set สำหรับแต่ละ Iteration

ทำการทดลองเปรียบเทียบค่า RMSE โดยใช้วิธี Mini Batch Gradient Descent โดยกำหนดจากการทำลอง ได้ผลลัพธ์ดังนี้ learning rate = 0.0005370044350549921, batch_size = 1000 และทำ 100 iteration แต่จะดูผลลัพธ์แค่ iteration 1 - 3 และ iteration 100 โดยได้ผลลัพธ์การทดลองดังนี้

Iteration = 1

RMSE from training set: 5.12471513729959

RMSE from test set 5.7265827506463225

Iteration = 2

RMSE from training set: 5.421225207102889

RMSE from test set 5.53031418637314

Iteration = 3

RMSE from training set: 4.875641942232389

RMSE from test set 5.195550981368578

. . . . . .

Iteration = 100

RMSE from training set: 1.4573251539838796

RMSE from test set 0.7866066361276137

พบว่า Iteration ที่ 1 - 3 นั้น ค่า RMSE ของ Training set และ Test set นั้น มีค่าอยู่ที่ 5 แต่เมื่อวนลูปอัปเดตค่า θ ไปเรื่อย ๆ ค่า RMSE ของ Training set ก็ลดลงเรื่อย ๆ ซึ่งค่าสุดท้าย Iteration ที่ 100 จะมีค่า RMSE ของ Training set อยู่ที่ 1.45 และค่า RMSE ของ Test set อยู่ที่ 0.78 ซึ่งสามารถอธิบายได้ว่า ยิ่งค่า RMSE ของ Training set ลดลง ค่า RMSE ของ Test set ก็ลดลงด้วยเช่นกัน ซึ่งหากทำการทดลองไปเรื่อย ๆ ค่า RMSE ของทั้ง Training set และ Test set จะลดลงไปเรื่อย ๆ เช่นกัน

2.4.2 ให้ออกแบบการทดลองเพื่อเปรียบเทียบค่าของ Squared Errors ของ Training set เมื่อมีการเปลี่ยนแปลงค่า Learning rate อภิปรายผลลัพธ์ที่ได้อย่างละเอียด

ได้มีการกำหนดค่า Learning Rate ไว้ดังนี้ learning_rates = [1, 0.1, 0.01, 0.001, 0.0001] จากนั้นทำการวนลูป ใน model ด้วยค่า learning_rates ดังกล่าว โดยมีการกำหนดค่า epochs ไว้เท่ากับ 10 และกำหนด method = 2 (ใช้วิธี Mini Batch Gradient Descent) และ batch_size = 100 ซึ่งได้ผลลัพธ์ดังนี้

At learning rate = 1 Squared Errors = [301643012082.0, 6.498638887062277e+18, 1.0020141138831667e+26, 1.3037746725741762e+33, 1.381470070106963e+40, 2.2185746811160015e+47, 2.0438984832517385e+54, 2.0736736270897545e+61, 2.3341265299646427e+68, 2.437976265100365e+75]

At learning rate = 0.1 Squared Errors = [2750852493.0, 271831993216099.0, 5.197900338411581e+19, 8.304338469835542e+24, 8.116429655832306e+29, 4.773974604838848e+34, 9.899319701812385e+39, 1.2862615944489677e+45, 1.6766276512607533e+50, 2.75255690110623e+55]

At learning rate = 0.01 Squared Errors = [24132092.0, 29901115316.0, 52427008097868.0, 7.20449634921851e+16, 9.864868546000277e+19, 9.14244109311977e+22, 8.969855572343081e+25, 8.636929898626863e+28, 1.4385349921512756e+32, 3.587841468357928e+35]

At learning rate = 0.001 Squared Errors = [210477.0, 2367301.0, 9874261.0, 82891293.0, 395257713.0, 2098108791.0, 7611000826.0, 34835566005.0, 167816051915.0, 1578811947364.0]

At learning rate = 0.0001 Squared Errors = [25499.0, 18472.0, 15206.0, 13824.0, 13234.0, 12720.0, 12579.0, 12383.0, 12110.0, 12042.0]

จะเห็นได้ว่าที่ค่า learning rate = 1, 0.1, 0.01, 0.001 นั้นเมื่ออัปเดตค่า θ แล้วนั้นส่งผลให้ค่า Squared Errors เพิ่มขึ้นเรื่อย ๆ ซึ่งหมายถึงค่า learning rate ไม่เหมาะสมกับข้อมูล ทำให้เมื่ออัปเดตค่า θ แล้วนั้น ส่งผลให้ Squared Errors เพิ่มขึ้นเรื่อย ๆ ในทางกลับกัน ที่ค่า learning rate = 0.0001 เมื่ออัปเดตค่า θ แล้วนั้นส่งผลให้ค่า Squared Errors ลดลงเรื่อย ๆ ซึ่งหมายความว่า learning rate นี้ เหมาะสมกับข้อมูล ทำให้เมื่ออัปเดตค่า θ แล้ว ส่งผลให้ Squared Errors ลดลงเรื่อย ๆ

สมาชิกกลุ่ม

นายภีมภัช พจน์สุนทร 6209650081

นางสาวนวพร วิริยะภาพ 6209650149

นางสาวพรกนก ศรีสังสิทธิสันติ 6209650214

นางสาวณัฐวรา บุญหนัก 6209650701