МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ

Магадлал статистик

Логистик регресс ашиглан орлогын түвшинг илрүүлэх

Шалгасан: Г. Махгал (Ахлах багш)

Гүйцэтгэсэн: Г. Жавхлан (22b1num3154)
Б. Балжинням (22b1num6983)
Т. Баасандорж (22b1num0004)
Б. Солонгоо (23b1num1034)

Улаанбаатар
2025 оны 12-р сарын 5

Агуулга

1 Оршил **3**

2 Өгөгдөл **3**
- 2.1 Өгөгдлийн эх үүсвэр ба зорилго . . . 3
- 2.2 Өгөгдлөө цэвэрлэх . . . 3
- 2.3 Өгөгдөл хуваах . . . 4
- 2.4 Урьдчилсан боловсруулалт . . . 4

3 Загварын хэрэгжүүлэлт **5**
- 3.1 Логистик регрессийн үндэс . . . 6
- 3.2 Сигмойд функц . . . 7
- 3.3 Эх олонгогийн үнэний хувь . . . 7
- 3.4 Үнэний хувиар алдааг илэрхийлэх . . . 7
- 3.5 Binary cross-entropy . . . 8
- 3.6 L2 тогтворжуулалт (regularization) . . . 8
- 3.7 Градиент бууруулалт (Gradient descent) . . . 9
- 3.8 Нэмэлт . . . 11
  - 3.8.1 Классын жин . . . 11
  - 3.8.2 Сургалтын эрчмийг багасгах . . . 11
  - 3.8.3 Early stopping . . . 11
  - 3.8.4 Заагийн утгыг оновчлох . . . 12
  - 3.8.5 Xavier/Glorot initialization . . . 12
- 3.9 Үндсэн fit() функц . . . 13

4 Үр дүн **14**
- 4.1 Ерөнхий гүйцэтгэл: . . . 14
- 4.2 Confusion матриц . . . 15
- 4.3 Онцлогийн ач холбогдол . . . 15
- 4.4 Магадлалын тархалт . . . 17

5 Дүгнэлт **17**

6 Багийн гишүүдийн оролцоо **18**

7 Ишлэл зүүлт **18**

1 Оршил

Энэ төслийн зорилго нь хувь хүний жилийн орлого 50,000 ам.доллароос дээш эсэхийг таамаглах явдал юм. Бид АНУ-ын Хүн амын тооллогын Adult Income өгөгдлийн санг ашиглан орлогын түвшинг урьдчилан таамаглах загвар боловсруулна. Энэхүү өгөгдлийн сан нь нас, боловсрол, мэргэжил, гэр бүлийн байдал зэрэг олон хувьсагчийг агуулдаг бөгөөд эдгээр нь хувь хүний санхүүгийн байдалд нөлөөлдөг гол хүчин зүйлс юм.

Төслийн үндсэн зорилго нь зөвхөн таамаглал гаргах бус, өгөгдөл дэх хамаарал, классын тэнцвэргүй байдал, оролцож буй хувьсагчдын нөлөөллийг ойлгож, загварын үйл ажиллагааг үнэлэхэд оршино. Энд бид логистик регрессийг ашиглан хоёртын ангиллын асуудлыг шийдвэрлэх ба загварын үзүүлэлтүүд нь орлогыг зөв таамаглах боломжийг хэр сайн хангаж байгааг илтгэнэ. Үүнээс гадна энэ төсөл нь өгөгдлийн шинжилгээ, ангиллын загварчлал болон статистик үндэслэлтэй шийдвэр гаргалтын практик дадлага олгоно.

2 Өгөгдөл

2.1 Өгөгдлийн эх үүсвэр ба зорилго

Бид Kaggle платформын Income Dataset буюу орлогын мэдээллийг ашигласан. Энэ өгөгдлийн багцад нас, боловсрол, мэргэжил, гэрлэлтийн байдал, хүйс зэрэг нийгэм эдийн засгийн шинж чанаруудыг илэрхийлэх олон хувьсагч орсон. Бидний зорилтот хувьсагч бол income_>50K (0 = $\leq 50K$, 1 = $> 50K$) юм. Өгөгдлийг цэвэрлэсний дараа үлдсэн хувьсагчдын мэдээллийг ашиглан энэ хувьсагчийн утгыг зөв таамаглах нь бидний үндсэн зорилго болно.

Түүврийн нийт хэмжээ нь ойролцоогоор 44,000 бөгөөд бид үүнийг сургах болон үнэлгээ хийх хоёр хэсэгт хуваана. Хуваалтын дараа 35165 мөр нь сургалтын хэсэг, 8792 мөр нь баталгаажуулалт, шалгалтын хэсгийг бүрдүүлнэ.

Гэвч өгөгдөлтэй ажиллах гол бэрхшээл нь классуудын тэнцвэргүй байдал юм. Ихэнх хүмүүс ($\approx$ 76%) $\leq 50K$ орлоготой. Харин цөөнх хувь нь ($\approx$ 24%) $> 50K$ орлоготой.

Өгөгдөлд хүмүүсийн ихэнх хувь нь 50K-аас их орлоготой байгаа тул энгийн загвар ашиглавал accuracy нь 76% гарна гэсэн үг. Гэвч энэ нь цөөнх 50K-аас бага орлоготой хүмүүсийг огт танихгүй тул хангалтгүй юм. Тиймээс бид 2 классыг хоёуланг нь оновчтойгоор авч үздэг логистик регрессийг ашиглаж, аль аль классыг нь зөв таамаглах боломжтой загвар боловсруулна. Үүний тулд F1 оноо, Recall зэрэг үзүүлэлтүүдийг чухалчилна.

2.2 Өгөгдлөө цэвэрлэх

Өгөгдлийн баганууд:

[age, workclass, fnlwgt, education, educational-num, marital-status, occupation, relationship, race, gender, capital-gain, capital-loss, hours-per-week, native-country, income_>50K]

Анхны өгөгдлийг шалгасны дараа бид давхардсан, ач холбогдол багатай, болон тайлбарлахад хэцүү багануудыг арилгаж, өгөгдлийг хялбаршуулсан.

Ашигласан хувьсагчид (9):

- Тоон хувьсагч (5): age, educational-num, capital-gain, capital-loss, hours-per-week
- Чанарын хувьсагч (4): education, marital-status, occupation, gender

Энэ нь загварын хурд, үр ашиг, тайлбарлах чадварыг нэмэгдүүлэх зорилготой.

2.3 Өгөгдөл хуваах

Бид өгөгдлийг сургалт болон баталгаажуулалт гэсэн хоёр хэсэгт **80/20** харьцаатайгаар хуваасан. Ингэхдээ income_>50K классын харьцаа тэнцвэртэй байхаар хуваасан бөгөөд үр дүнд нь дараах 2 файлыг үүсгэж хадгалсан:

- Сургалтын багц: train_split.csv
- Баталгаажуулалтын багц: val_split.csv

Анхны өгөгдлийн багц дотор income_>50K хувьсагчийн 76% нь 0, 24% нь 1 утгатай байсан бол хуваалтын дараа энэ харьцаа эвдрээгүй, хэвээрээ үлдсэн.

```
# Өгөгдлийг унших
df = pd.read_csv("data.csv")

# Ашиглах хувьсагчид
keep_cols = ["age", "educational-num", "capital-gain", "capital-loss",
             "hours-per-week", "education", "marital-status", "occupation",
             "gender", "income_>50K"]
df = df[keep_cols]

# Сургалт/баталгаажуулалт хуваалт
train_df, val_df = train_test_split(
    df, test_size=0.2, random_state=42, stratify=df["income_>50K"]
)

# Файлуудыг хадгалах
train_df.to_csv("train_split.csv", index=False)
val_df.to_csv("val_split.csv", index=False)
```

2.4 Урьдчилсан боловсруулалт

Логистик регресс загварт өгөгдлийг оруулахын өмнө тоон болон категори хувьсагчийг зохих хэлбэрт хөрвүүлэх шаардлагатай. Энэ процессыг дараах байдлаар хийсэн.

Тоон хувьсагчид:

StandardScaler ашиглаж тоон хувьсагч бүрийн дундаж утгыг 0, стандарт хазайлтыг 1 болгож нормальчилсан. Учир нь логистик регресс нь градиент дээр суурилсан алгоритм тул хувьсагчийн тархалт, хэмжээнээс хамаарч удаан суралцаж, тогтворгүй байж болно.

Иймд хувьсагчийг түүврийн дундаж руу нь төвлөрүүлснээр сургалтын эрчим, тогтвортой байдал нэмэгдэнэ.

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

Чанарын хувьсагчид:

OneHotEncoder ашиглаж чанарын хувьсагч бүрийг хоёртын вектор болгон хөрвүүлсэн. Учир нь машин сургалтын алгоритмууд текст буюу категори утгыг шууд ойлгох боломжгүй тул категори утгыг тоон хэлбэрт хөрвүүлэх шаардлагатай байдаг. Энэ нь ямар ч дараалал, зэрэглэлгүй тус бүрийн категори утгыг 0/1 утгаар илэрхийлдэг.

Жишээлбэл, education гэх чанарын хувьсагчийг авч үзье. Үүний утгууд нь:

$[E = \{\text{HS}, \text{Bachelors}, \text{Masters}, \text{Doctorate}\}]$

One-hot кодчилол нь уг хувьсагчийг дараах байдлаар хувиргана:

$$f(x) = \begin{cases} [1, 0, 0, 0] & \text{if } x = \text{HS} \\ [0, 1, 0, 0] & \text{if } x = \text{Bachelors} \\ [0, 0, 1, 0] & \text{if } x = \text{Masters} \\ [0, 0, 0, 1] & \text{if } x = \text{Doctorate} \end{cases}$$

Энэ нь тус бүрийн категори хувьсагчийг Бернуллийн хувьсагч болгож байна гэсэн үг юм.

Загварын гиперпараметр:

- learning_rate ойролцоогоор 0.1
- max_iter ойролцоогоор 1000
- reg_lambda = 1e-4 (L2)
- lr_decay = 1e-4
- threshold = 0.5 (шийдвэрийн хязгаар)

3 Загварын хэрэгжүүлэлт

Бид энэ төсөлд логистик регрессийн загварыг гараар хэрэгжүүлсэн. Яагаад гэвэл sklearn-ийн LogisticRegression нь олон зүйлийг автоматаар хийдэг бөгөөд бид хэрхэн ажилладгийг нь ойлгохыг илүүд үзлээ. Мөн сурах эрчмийн бууралт, class weighting зэрэг сонирхолтой зүйлсийг өөрсдөө туршиж үзэхийг хүссэн.

3.1 Логистик регрессийн үндэс

Логистик регресс нь хоёртын ангилал хийх суурь загваруудын нэг юм. Шугаман регресс нь тасралтгүй утгуудыг таамагладаг бол логистик регресс нь аливаа инстанц нь тодорхой классын гишүүн байх магадлалыг тооцоолдог.

Оролтын хувьсагч x -н хувьд

$$P(y = 1 \mid x)$$

буюу гаралт y нь 1-тэй тэнцүү байх магадлалыг олох зорилготой.

Энэ төслийн хувьд бидний зорилтод хувьсагч маань income > 50K үед 1, income $\leq \quad 50K$ үед 0 гэсэн хоёр л утга авна. Иймээс түүвэр болгоны хувьд энэ хувьсагчийг санамсаргүй хувьсагчаар загварчилж болно.

$$y_i \sim \text{Bernoulli}(p_i)$$

Энд

- y_i = i-р түүвэр дээр ажиглагдсан утга
- p_i = $P(y_i = 1 \mid x_i)$ = i-р түүвэр класс 1-ийн гишүүн байх магадлал

Нэг түүврийн тархалт нь:

$$P(y_i \mid p_i) = p_i^{y_i}(1 - p_i)^{1-y_i}.$$

Одоо бид гарт оролтын онцлог x_i-с класс 1-ийн магадлалыг илэрхийлэх функц хэрэгтэй байна.

Эхлээд бид оролтуудын шугаман тэгшитгэлийг бодно:

$$z_i = w^\top x_i + b$$

Энд:

- w = жингийн вектор (сурах параметрүүд)
- b = хазайлтын утга
- x_i = онцлогийн вектор i

```
if issparse(X_array):
    z = X_array.dot(self.weights) + self.bias
else:
    z = np.dot(X_array, self.weights) + self.bias
```

3.2 Сигмойд функц

Гэхдээ дээрх шугаман тэгшитгэл нь ямар ч жинхэнэ утгыг өгөх боломжтой. Бид гарсан тэр жинхэнэ утгыг $[0, 1]$ хооронд орших магадлал болгох хэрэгтэй. Үүний тулд доорх сигмойд функцээр илэрхийлэгдэх логистик муруйг ашиглана.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

```python
def sigmoid(self, z):
    return 1 / (1 + np.exp(-np.clip(z, -500, 500)))  # overflow-оос сэргийлэх
```

Эндээс модел нэг түүврийн хувьд таамаглах магадлал нь дараах томъёогоор илэрхийлэгдэнэ:

$$\hat{p}_i = P(y_i = 1 \mid x_i) = \sigma(z_i) = \sigma(w^\top x_i + b)$$

3.3 Эх олонгогийн үнэний хувь

Бүх y_i нь Бернуллийн санамсаргүй утга. Өгөгдлийн цэгүүд нь бусдаасаа хамааралгүй гэж үзвэл w, b хоёр параметрүүдийг өгсөн үед бүх y_i-г ажиглах магадлал нь:

$$\mathcal{L}(w, b) = \prod_{i=1}^{m} \hat{p}_i^{y_i} (1 - \hat{p}_i)^{1-y_i}$$

Энд:

- y = жинхэнэ утга (0 or 1)
- $\hat{p}$ = таамагласан утга

Үнэний хувийн функц нь w, b хоёр параметрүүдийг өгсөн үед бидний эх олонлог ажиглагдах магадлал хэд вэ гэдгийг олж болно гэсэн үг. Энд хамгийн их үнэний хувь бүхий үнэлгээг (maximum likelihood estimation) хийснээр хамгийн оновчтой параметрүүдийг олно.

3.4 Үнэний хувиар алдааг илэрхийлэх

Хэрвээ бид үржвэрүүдтэй ажиллах бол компьютер дүрслэх боломжгүй жижиг тоотой ажиллах хэрэгтэй болох тул үнэний хувийн функц $\mathcal{L}(w, b)$-н натурал логарифмийг авна.

$$\log \mathcal{L}(w, b) = \sum_{i=1}^{m} [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)]$$

Энийг алдааны функц болгохын тулд -1-р үржүүлнэ. Өөрөөр хэлбэл binary cross-entropy loss фунцкээ гаргаж ирлээ.

$$L_{\mathsf{CE}}(w,b) = -\sum_{i=1}^{m} \left[y_i \log(\widehat{p}_i) + (1-y_i)\log(1-\widehat{p}_i) \right]$$

3.5 Binary cross-entropy

Нэг түүврийн алдааг олох томъёо:

$$\ell = -\left[y \log(\widehat{p}) + (1-y)\log(1-\widehat{p}) \right].$$

буюу:

- $if\ y = 1 \quad \ell = -\log(\widehat{p})$
- $if\ y = 0 \quad \ell = -\log(1-\widehat{p})$

Cross-entropy нь загварыг итгэлтэйгээр буруу таамаглал гаргахыг илүү шийтгэдэг. Өөрөөр хэлбэл 1 эсвэл 0-тэй маш ойрхон магадлал (0.99, 0.01 г.м.) гаргаад энэ нь буруу болж таарвал алдаа нь өндөр гарч ирнэ.

Таамаглал $\widehat{p}$	Алдаа $-\log(\widehat{p})$	Шийтгэлийн түвшин
0.99 (итгэлтэй, зөв)	0.01	Маш бага
0.50 (итгэл багатай)	0.69	Дунд зэргийн
0.10 (итгэлтэй, буруу)	2.30	Том
0.01 (маш итгэлтэй, буруу)	4.61	Маш том

$$L_{\mathsf{CE}} = -\frac{1}{m}\sum_{i=1}^{m} \left[y_i \log(\widehat{p}_i) + (1-y_i)\log(1-\widehat{p}_i) \right].$$

3.6 L2 тогтворжуулалт (regularization)

Хэт том утгатай жингээс үүдэлтэй overfitting-ээс сэргийлнэ

$$\frac{\lambda}{2m}\|w\|^2$$

Энд:

- λ = тогтворжуулалтын хүч (гиперпараметр)
- m = түүврийн хэмжээ
- $\|w\|^2 = w_1^2 + w_2^2 + \cdots + w_n^2$

Нийт алдаа:

$$L = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log(\hat{p}_i) + (1-y_i)\log(1-\hat{p}_i)\right] + \frac{\lambda}{2m}\|w\|^2$$

```
def compute_loss(self, y_true, y_pred, sample_weights=None):

    m = len(y_true)

    epsilon = 1e-15
    y_pred = np.clip(y_pred, epsilon, 1 - epsilon)

    sample_losses = -(y_true * np.log(y_pred) +
                     (1 - y_true) * np.log(1 - y_pred))

    if sample_weights is not None:
        sample_losses = sample_losses * sample_weights

    cross_entropy = np.mean(sample_losses)

    l2_penalty = (self.reg_lambda / (2 * m)) * np.sum(self.weights ** 2)

    return cross_entropy + l2_penalty
```

3.7 Градиент буурулалт (Gradient descent)

Одоо манай алдагдлын функц тодорхойлогдсон. Гэхдээ нийт алдааг багасгахын тулд градиент буурулалт гэж нэрлэгддэг аргыг ашиглана. Энэ тохиолдолд градиент гэдэг нь жингүүдийн аль чиглэлд өөрчлөгдөх үед алдаа нь хамгийн хурдтай өсөж буурахыг илэрхийлдэг вектор.

Градиент буурулалтын үндсэн санаа нь уг градиентийг тооцоолоход оршино.

Жингийн градиент:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{m}X^\top(\hat{p} - y) + \frac{\lambda}{m}w$$

- $X \in \mathbb{R}^{m\times n}$: онцлогийн матриц ($m \times n$)
- $w \in \mathbb{R}^{n\times 1}$: жингийн вектор
- $\hat{p} \in \mathbb{R}^{m\times 1}$: түүвэр болгоны таамагласан утга
- $y \in \mathbb{R}^{m\times 1}$: жинхэнэ утгууд
- λ: тогторжуулалтын хүч

Жингийн градиент нь $n \times 1$ вектор.

$$\frac{\partial \mathcal{L}}{\partial w} \in \mathbb{R}^{n \times 1}$$

Хазайлтын градиент:

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (\hat{p}_i - y_i)$$

Жинг өөрчлөх дүрэм:

$$w \leftarrow w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$

$$b \leftarrow b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$

α = сурах эрчим. α нь тогтмол утга бөгөөд нэг сургалтын итераци болгонд жин, хазайлт хоёрын өөрчлөлтийг удирдах коэффициент.

```
def _compute_gradients(self, X, y, y_pred):
    m = len(y_true)
    # error
    error = y_pred - y_true

    if sample_weights is not None:
        error = error * sample_weights

    # Жингийн градиент
    if issparse(X):
        dw = (1/m) * X.T.dot(error)
        dw = np.asarray(dw).flatten()
    else:
        dw = (1/m) * np.dot(X.T, error)

    # + L2
    dw += (self.reg_lambda / m) * self.weights.flatten()

    # Bias-ийн градиент
    db = (1/m) * np.sum(error)

    return dw, db
```

Сургалт нь дараах нөхцлийн аль нэг нь биелэхэд зогсоно: 1. Сургалтын алдаа тодорхой утга руу нийлэх үед 2. max_iter 1000 давах үед 3. Баталгаажуулалтын алдаа нь багасахаа болих үед (early stopping)

3.8 Нэмэлт

3.8.1 Классын жин

Бидний өгөгдөл нь классын хувьд тэнцвэргүй байсан тул $c \in \{0,1\} : w_c = \frac{m}{2 \cdot m_c}$ зарчмаар классын жинг тооцоолсон. Ингэснээр манай загвар цөөнх класс дээр илүү их ач холбогдол өгнө.

```
self.class_weights_ = self.compute_class_weights(y.flatten())
sample_weights = np.array([self.class_weights_[cls] for cls in y.flatten()]).reshape(-1, 1)
```

- m: түүврийн хэмжээ
- m_c: класс {c} -ийн түүврийн хэмжээ

Түүвэр болгоны алдаа нь классынх нь жингээр үржигднэ:

$$\ell_i = -w_{y_i} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

3.8.2 Сургалтын эрчмийг багасгах

Сургалтын эрчим нь сургалтын t-р давталт дээр дараах зарчмаар багасна:

$$\alpha_t = \frac{\alpha_0}{1 + \text{decay} \cdot t}$$

- α_0: анхны сургалтын хэмжээ
- $decay$: тогтмол, гиперпараметр

```
self.learning_rate = self.initial_lr / (1 + self.lr_decay * iteration)
```

Бидний моделийн градиент бууруулалтын дүрэм:

$$w \leftarrow w - \alpha_t \frac{\partial \mathcal{L}}{\partial w}, \quad b \leftarrow b - \alpha_t \frac{\partial \mathcal{L}}{\partial b}$$

3.8.3 Early stopping

Алдаа нь баталгаажуулалтын өгөгдөл дээр багасахаа болих үед сургалтыг зогсооно. ``Overfitting'' тохиолдохоос сэргийлэх арга.

```
if val_loss < best_val_loss:
    best_val_loss = val_loss
    patience_counter = 0
else:
    patience_counter += 1
    if patience_counter >= patience:
        break
```

3.8.4 Заагийн утгыг оновчлох

Precision, recall, эсвэл F1 үзүүлэлтийг дээд зэргээр ихэсгэх заагийн утгыг олно. Энэ утгыг $[0.1, 0.9]$ интервалаас хайна.

```
def optimize_threshold(self, X, y_true, metric='f1'):
    probas = self.predict_proba(X)[:, 1]
    thresholds = np.linspace(0.1, 0.9, 81)

    best_score = 0
    best_threshold = 0.5

    for threshold in thresholds:
        y_pred = (probas >= threshold).astype(int)

        if metric == 'f1':
            score = f1_score(y_true, y_pred)
        elif metric == 'precision':
            score = precision_score(y_true, y_pred)
        elif metric == 'recall':
            score = recall_score(y_true, y_pred)
        else:
            score = f1_score(y_true, y_pred)

        if score > best_score:
            best_score = score
            best_threshold = threshold

    self.threshold = best_threshold
    return best_threshold
```

3.8.5 Xavier/Glorot initialization

Онцлогуудын жингийн вектороо 0 утгатай зарлахын оронд $[1; n]$ хүртэлх санамсаргүй утгуудаар дүүргээд $\frac{1}{n}$-р үржүүлнэ. numpy.random.randn() функц нь стандарт хэвийн тархалтаас түүврүүдээ үүсгэдэг. $\mu = 0, \sigma^2 = 1$. $\frac{1}{n}$-р үржүүлснээр $\sigma^2 = \frac{1}{n}$ болох ба энэ нь градиент буурруулалт болон бусад машин сургалтын алгоритмуудын үр ашгийг нэмэгдүүлнэ. Градиентийг хэт их/бага утга руу тэмүүлэхээс сэргийлнэ.

```
self.weights = np.random.randn(n, 1) * np.sqrt(1.0 / n)
```

3.9 Үндсэн fit() функц

```
def fit(self, X, y, X_val=None, y_val=None):
    if issparse(X):
        m, n = X.shape
        X_array = X
    else:
        X = np.array(X)
        m, n = X.shape
        X_array = X

    y = np.array(y).reshape(-1, 1)

    self.class_weights_ = self.compute_class_weights(y.flatten())
    sample_weights = np.array([self.class_weights_[cls] for cls in y.flatten()]).reshape(-1, 1)

    self.weights = np.random.randn(n, 1) * np.sqrt(1.0 / n)
    self.bias = 0

    prev_loss = float('inf')
    best_val_loss = float('inf')
    patience_counter = 0
    patience = 10

    for iteration in range(self.max_iter):
        self.learning_rate = self.initial_lr / (1 + self.lr_decay * iteration)

        if issparse(X_array):
            z = X_array.dot(self.weights) + self.bias
            z = np.asarray(z).flatten().reshape(-1, 1)
        else:
            z = np.dot(X_array, self.weights) + self.bias

        y_pred = self.sigmoid(z)

        loss = self.compute_loss(y, y_pred, sample_weights)
        self.losses.append(loss)

        dw, db = self.compute_gradients(X_array, y, y_pred, sample_weights)
```

```
            self.weights -= self.learning_rate * dw.reshape(-1, 1)
            self.bias -= self.learning_rate * db

            if X_val is not None and y_val is not None:
                val_loss = self._compute_val_loss(X_val, y_val)
                if val_loss < best_val_loss:
                    best_val_loss = val_loss
                    patience_counter = 0
                else:
                    patience_counter += 1
                    if patience_counter >= patience:
                        break

            if abs(prev_loss - loss) < self.tol:
                break
            prev_loss = loss

        return self
```

4 Үр дүн

4.1 Ерөнхий гүйцэтгэл:

Манай загвар эцэст нь хамааран хувьсагч y_i-н утгыг зөв таахад л оршино. Энэ хувьсагч нь $\{0, 1\}$ гэсэн хоёрхон утга авах ба энэ нь түүврийн класс болно. Бид баталгаажуулалтын өгөгдөл дээр 0 классын түүвэрт зөв таамаглал гаргах тоолонд true negative, харин буруу гаргахад false negative үзүүлэлт 1-ээр нэмэгднэ. Мөн адил 1 класс дээр зөв таах нь true positive, алдах нь false positive үзүүлэлтүүд нэмэгднэ.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

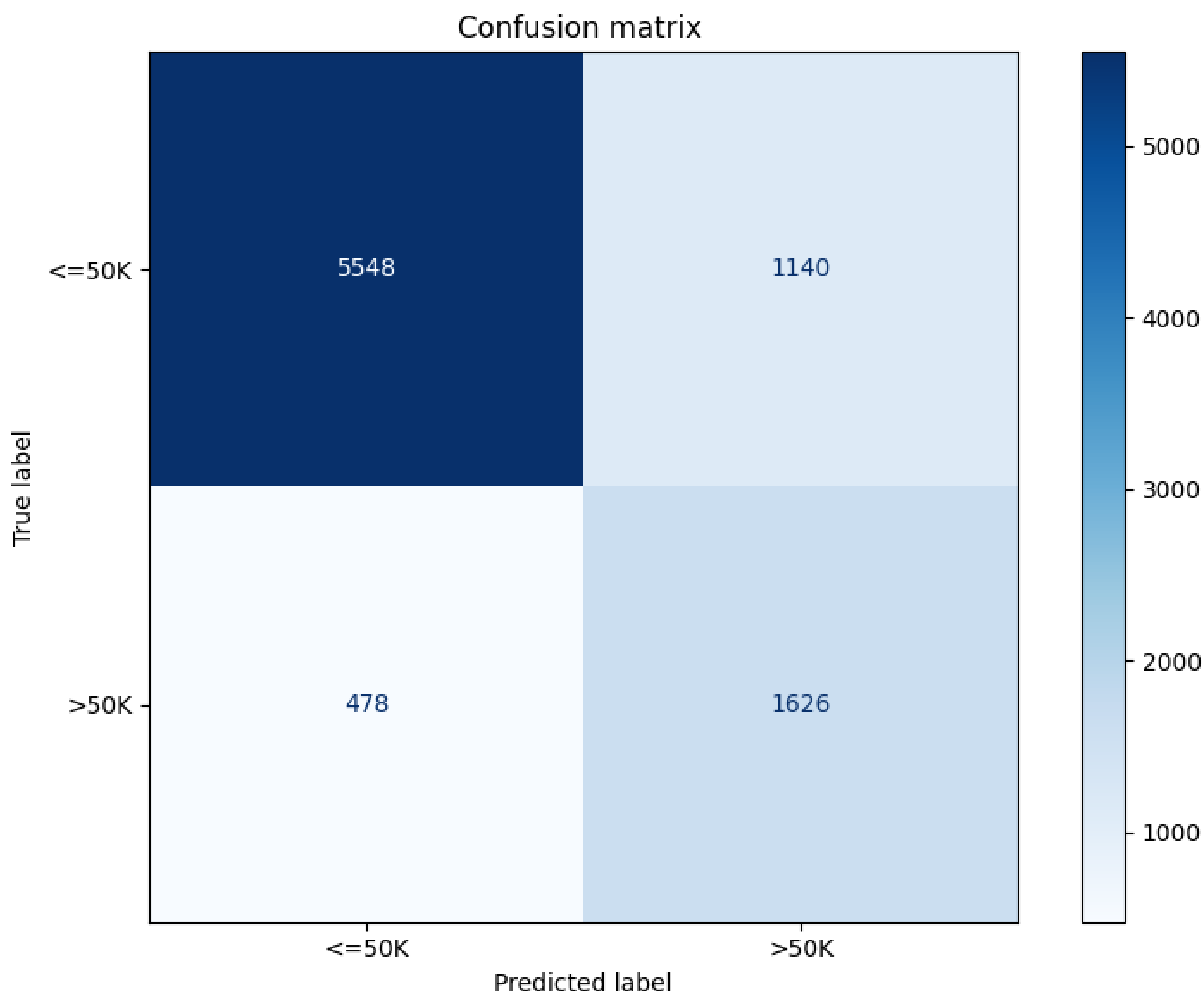
Моделын заагийн утгыг F1 утгын хувьд оновчлох үед ерөнхий үзүүлэлтүүд:

- **Accuracy**: $\approx$ 0.82

- **Precision**: ≈ 0.60
- **Recall**: ≈ 0.74
- **F1 Score**: ≈ 0.66

Бидний модел класс 1-н түүврүүдийн хувьд ихэнхдээ зөв таамаглал гаргадаг ч precision = 60% байгаа нь 1 гэж таамагласан түүврүүдийн 40% нь 0 байсан гэдгийг илтгэж байна. Мөн recall = 74% байгаа нь нийт класс 1-н түүврүүдийн 74%-г зөв таасан гэдгийг харууллаа.

4.2 Confusion матриц



Зураг 1: Confusion матриц

4.3 Онцлогийн ач холбогдол

Сигмоид функц шугаман нийлбэрийг ($z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$) магадлал руу хөрвүүлдэг.

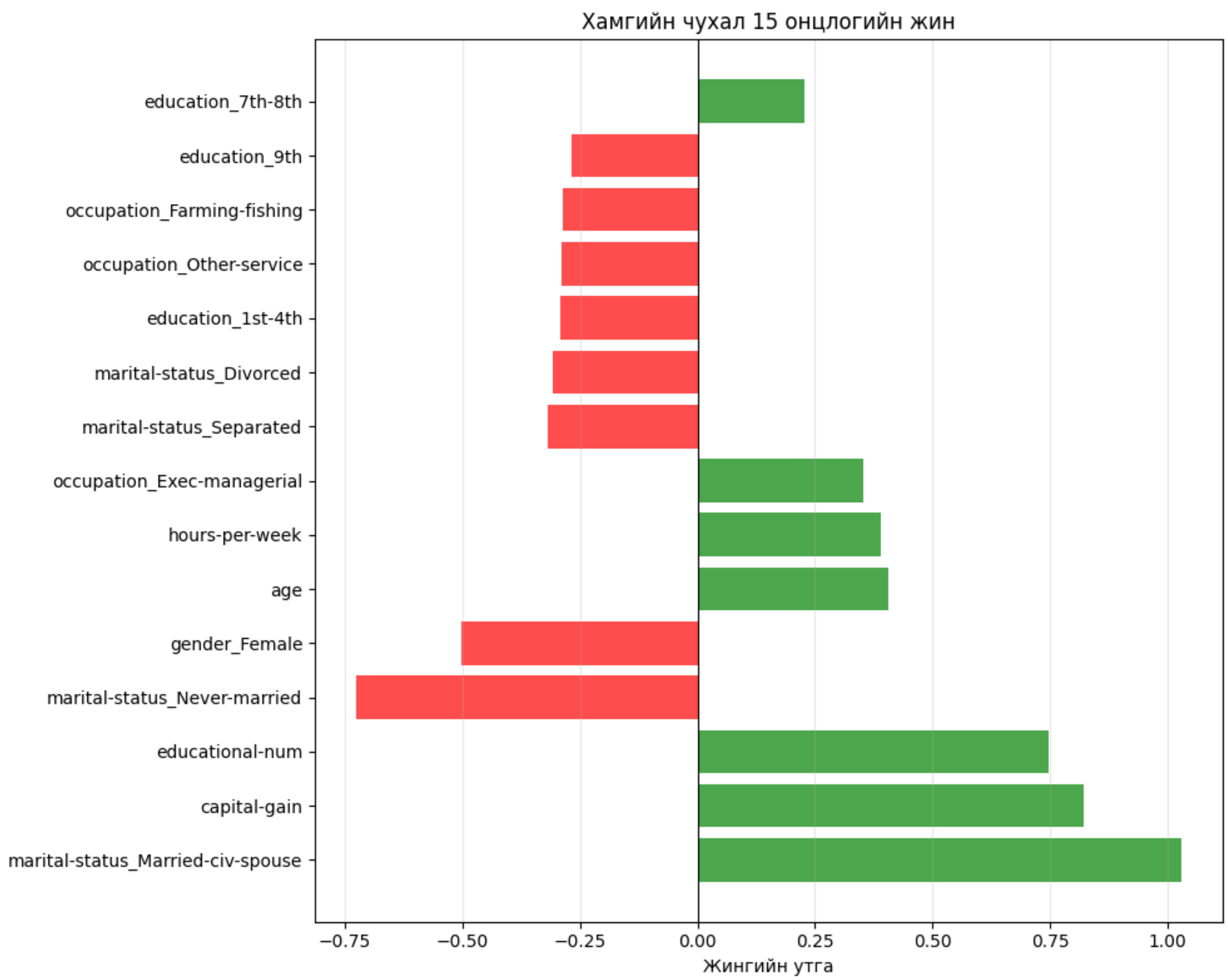
- $z \to +\infty$, үед $e^{-z} \to 0$, учир:

$$\sigma(z) \to \frac{1}{1+0} = 1$$

- Харин $z \to -\infty$, үед $e^{-z} \to \infty$, учир:

$$\sigma(z) \to \frac{1}{1+\infty} = 0$$

Жингийн утга нь ихсэх тусам тухайн түүвэр нь класс 1-н гишүүн байх магадлал нь дагаж ихэснэ. Жингүүдийг харахад аль онцлог x хамгийн чухал болохыг харж болно.

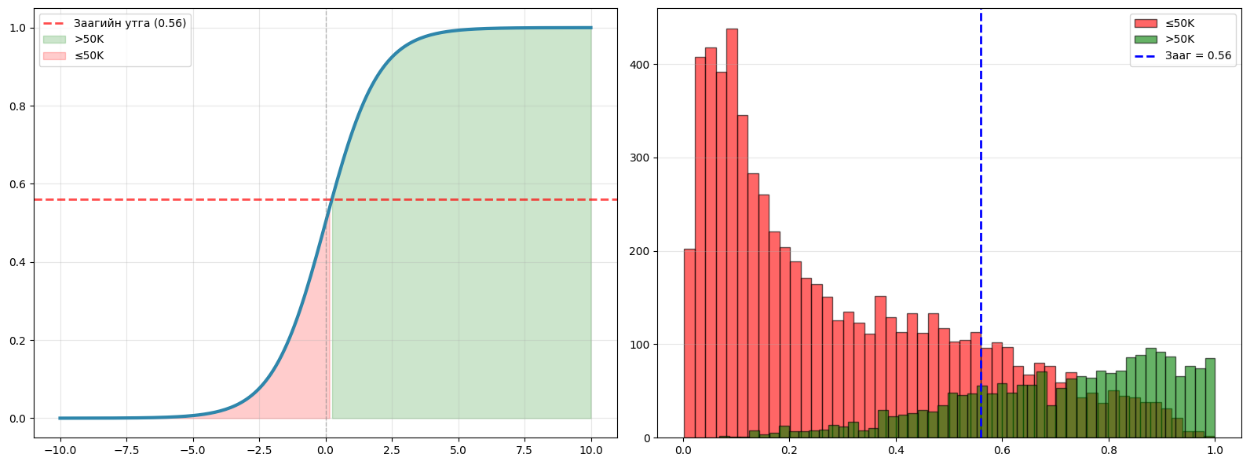


Зураг 2: Хамгийн чухал 15 онцлогийн жин

4.4 Магадлалын тархалт

Зарим >50K орлоготой хүмүүс өндөр магадлалтай (0.9+) гарч байгаа ч, зарим нь харьцангүй бага магадлалтай (0.2--0.4) байна. Энэ магадлалын давхцал нь зарим өгөгдөл дээр тодорхой ангилалт хийх бэрхшээлийг харуулж байна.

Сигмоид функц нь загварын гаргасан тоон үнэлгээг (score) магадлал руу хөрвүүлдэг. Зүүн талд математик функц, баруун талд бидний загварын бодит таамаглалууд хэрхэн тархсаныг харуулж байна:



Зураг 3: Сигмоид функц ба магадлалын тархалт

Зүүн талын график нь онолын хэсэг --- $z = 0$ үед магадлал яг 0.5 байна. Харин баруун талын гистограм нь бодит байдлыг харуулна:

- **Улаан хэсэг ($\leq 50K$):** Ихэнх нь $0 - 0.3$ магадлалтай байна. Загвар энэ хүмүүсийг бага орлоготой гэдэгтээ нэлээд итгэлтэй байна.
- **Ногоон хэсэг ($> 50K$):** Тархалт нь 0.2 - оос 0.9 хүртэл маш өргөн байна. Энд давхцал их байгааг анзаараарай.

5 Дүгнэлт

Энэхүү төсөл нь логистик регрессийг практикт хэрэгжүүлэх явцдаа зөвхөн алгоритмын ажиллагаа төдийгүй өгөгдлийн чанар, статистик ойлголтууд загварын гүйцэтгэлд ямар их нөлөөтэйг бодитоор мэдрэх боломж олголоо. Загвар тогтвортой суралцаж, overfitting ажиглагдаагүй нь L2 арга болон градиент буурулалт оновчтой байсны илрэл юм. Гэсэн хэдий ч гүйцэтгэл, ялангуяа F1 үнэлгээ нь 0.66 давахгүй байгаа нь өгөгдлийн бүтэц, классын тэнцвэргүй байдал зэрэгтэй холбоотой.

Ирээдүйд Random forest, decision tree зэрэг шугаман бус загваруудыг ашиглавал >50K орлоготой хүмүүсийг илүү найдвартай таамаглах боломжтой. Гол сургамж нь зөвхөн accuracy-аас гадна precision, recall, F1 үзүүлэлтүүдийг ойлгон, ашиглах загвараа болон өгөгдлийн чанарыг хамтад нь үнэлэх хэрэгтэй гэдгийг харуулж байна.

6 Багийн гишүүдийн оролцоо

Багаараа цаг товлон уулзаж, зорилго болон сэдэв сонголтоо хийсэн. Хүн бүр өөрсдийн хийх ажлыг хуваан авч гэр гэрээсээ хийхээр болсон. Github дээр шинэ төсөл үүсгэж 4 гишүүд бүгд жигд оролцон хийсэн.

7 Ишлэл зүүлт

- Scikit□learn developers. (n.d.). *Logistic Regression --- scikit□learn documentation*. Retrieved from https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
- StatQuest. (n.d.). *Logistic Regression* [YouTube playlist]. Retrieved from https://www.youtube.com/watch?v
- Russell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach* (3rd ed.).
- Google Developers. (n.d.). *Logistic regression: Loss and regularization*. Retrieved from https://developers.google.com/machine-learning/crash-course/logistic-regression/loss-regularization :contentReferenceoaicite:0
- Danushka. (n.d.). *Logistic Regression (lecture notes/pdf)*. Retrieved from https://danushka.net/lect/dm/logre :contentReferenceoaicite:1
- Curtis, F. E., & Scheinberg, K. (2017). *Optimization Methods for Supervised Machine Learning: From Linear Models to Deep Learning*. arXiv preprint arXiv:1706.10207. :contentReferenceoaicit