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

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.

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

```
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.

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

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

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

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

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

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:

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

```
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 бууруулалт (Gradient descent)

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

Градиент бууруулалтын 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: тогворжуулалтын 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 бууруулалт 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
```

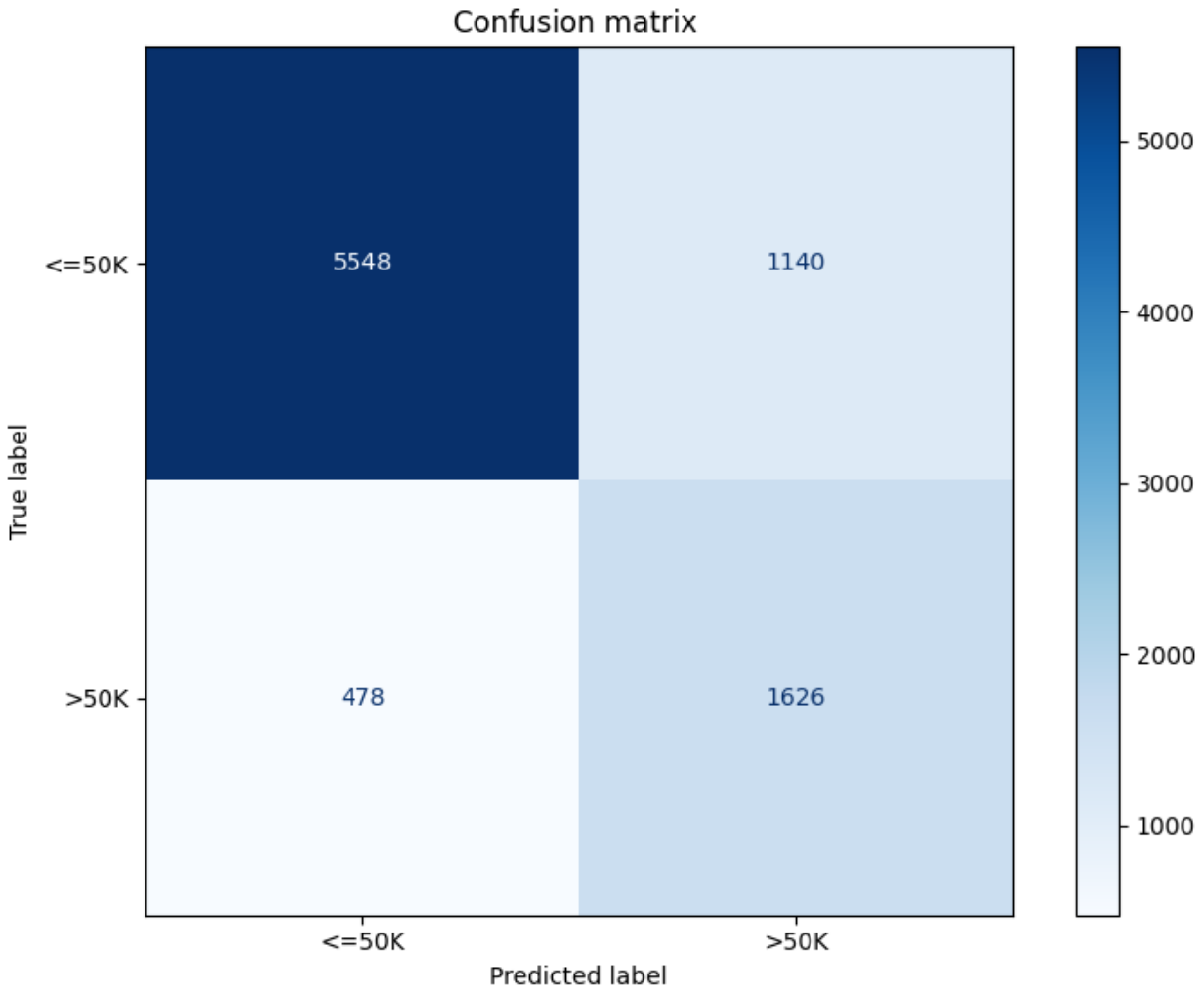
4 Үр дүн

4.1 Ерөнхий гүйцэтгэл:

Эдгээр тоонууд юу гэсэн үг вэ? Бидний загвар ерөнхийдөө сайн ажилладаг боловч цөөнх классыг (>50K) таних нь илүү хэцүү байна. Энэ нь тэнцвэргүй өгөгдлийн ердийн асуудал юм.

4.2 Confusion матриц

Confusion матриц нь манай алдаануудын төрлийг харуулна:



Зураг 1: Confusion матриц

Яагаад вэ? Өгөгдөл тэнцвэргүй байгаа учраас, энэ загвар байнга олонх классыг ($\leq \quad 50K$) харахаар сургагдсан. Энэ нь цөөнх классыг таних нь хэцүү болгодог.

4.3 Класс тус бүр дээрх гүйцэтгэл

Класс тус бүрээр нарийвчлан харвал:

$\leq 50K$ **анги (олонх):** - Precision: 0.89 - Recall: 0.91
- F1: 0.90

Загвар энэ классыг амархан таньдаг.

$> 50K$ **анги (цөөнх):** - Precision: 0.66 - Recall: 0.59 - F1: 0.62

Илүү муу. Энэ нь тэнцвэргүй өгөгдлийн шууд үр дагавар --- цөөн дээжтэй ангиудыг суралцах нь хэцүү.

4.4 Онцлогийн ач холбогдол

Жингүүдийг харахад, юу хамгийн чухал болохыг харж болно. Эхний 5 эерэг ба сөрөг онцлогууд:

Эерэг нөлөө (>50K рүү ойртуулах):

1. **educational-num (+0.89):** Боловсрол өндөр байх тусам орлого өндөр байх магадлал ихтэй.
2. **capital-gain (+0.76):** Хэрэв та хөрөнгө оруулалтаас орлого олж байвал орлого өндөр байх магадлал дээшилнэ.
3. **marital-status_Married-civ-spouse (+0.46):** Гэрлэсэн хүмүүс илүү тогтвортой орлоготой байх хандлагатай.
4. **occupation_Exec-managerial (+0.52):** Удирдах албан тушаал = илүү өндөр цалин.
5. **hours-per-week (+0.28):** Илүү их ажилласан = илүү их мөнгө.

Сөрөг нөлөө ($\leq 50K$ рүү ойртуулах):

1. **occupation_Other-service (-0.67):** Үйлчилгээний ажил ихэвчлэн бага цалинтай.
2. **marital-status_Never-married (-0.48):** Ганц бие байх нь бага орлоготой холбоотой (гэхдээ бодит байдалд бусад хүчин зүйлээс шалтгаалж магадгүй).
3. **capital-loss (-0.41):** Хөрөнгийн алдагдал санхүүгийн асуудлын шинж тэмдэг.
4. **education_HS-grad (-0.32):** Бүрнэ дунд боловсрол дангаараа өндөр орлого хангахад хүрэлцэхгүй.

4.5 Магадлалын тархалт

Сигмоид функц шугаман нийлбэр буюу linear combination-ийг ($z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$) магадлал руу хөрвүүлдэг.

$\leq 50K$ **хүмүүсийн хувьд:** - Дундаж магадлал (>50K байх): **0.18** - Ихэнх нь 0-0.3 хооронд

Загвар эдгээр хүмүүсийг 100% эерэг биш гэж бодож байна, гэхдээ ихэвчлэн магадлалыг бага байлгадаг.

$> 50K$ **хүмүүсийн хувьд:** - Дундаж магадлал: **0.57** - Илүү өргөн тархалттай (0.2-0.9)

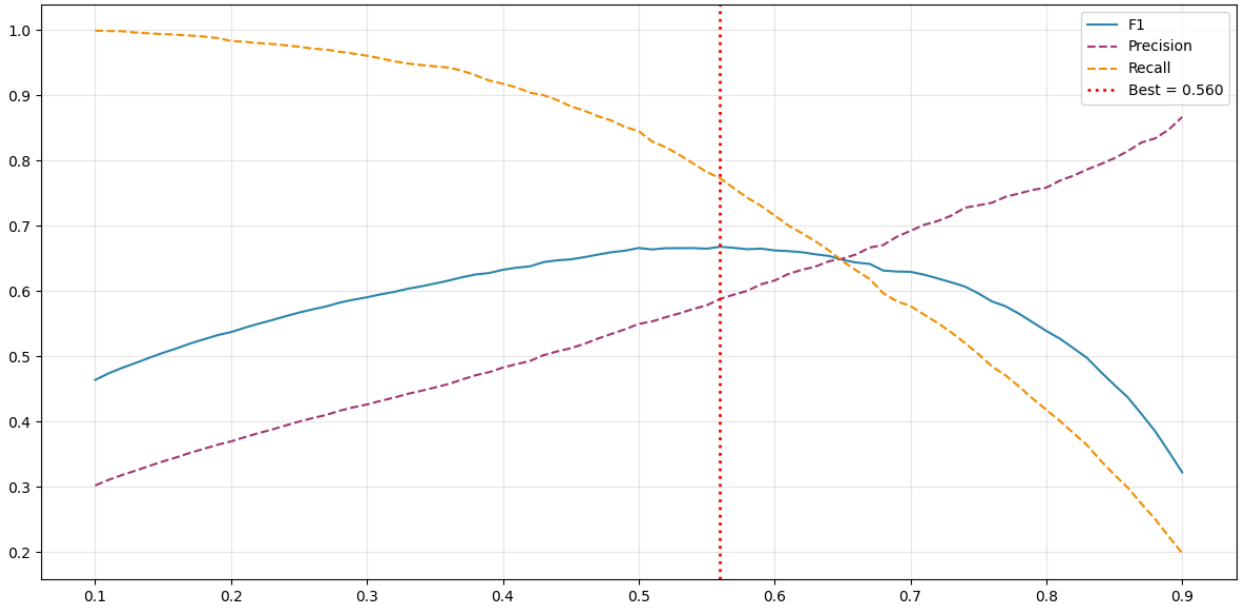
Зарим >50K орлоготой хүмүүс өндөр магадлалтай (0.9+) гарч байгаа ч, зарим нь харьцангуй бага магадлалтай (0.2--0.4) байна. Энэ магадлалын давхцал нь загвар төгс ажиллахаас өөр аргагүйг харуулж байна. Зарим өгөгдөл дээр тодорхой ангилалт хийх хэцүү.

4.6 PR Trade-off

Яагаад бид 100% precision ба 100% recall-д зэрэг хүрч чадахгүй гэж?

Математик талаас боломжгүй, учир нь ангиуд заримдаа давхцдаг. Манай магадлалын тархалтын график үүнийг тодорхой харуулна. Зарим өндөр орлоготой хүмүүсийг загвар бага магадлалтай гэж үзэж, зарим $\leq \quad 50K$ хүмүүс өндөр магадлалтай гэж үнэлдэг. Иймээс бид аль ч загвараар 100% зөв ангилах боломжгүй.

Загварын ерөнхий гүйцэтгэлийг ROC AUC=0.891 үзүүлж байна. Энэ нь санамсаргүй таамаглалаас (0.5) илүү сайн бөгөөд ангилалд дунджаар үнэлгээ сайн байгааг харуулна. Төгс үзүүлэлт биш (1.0), гэхдээ оюутны төсөлд анхны оролдлого болгон маш сайн гүйцэтгэл юм.



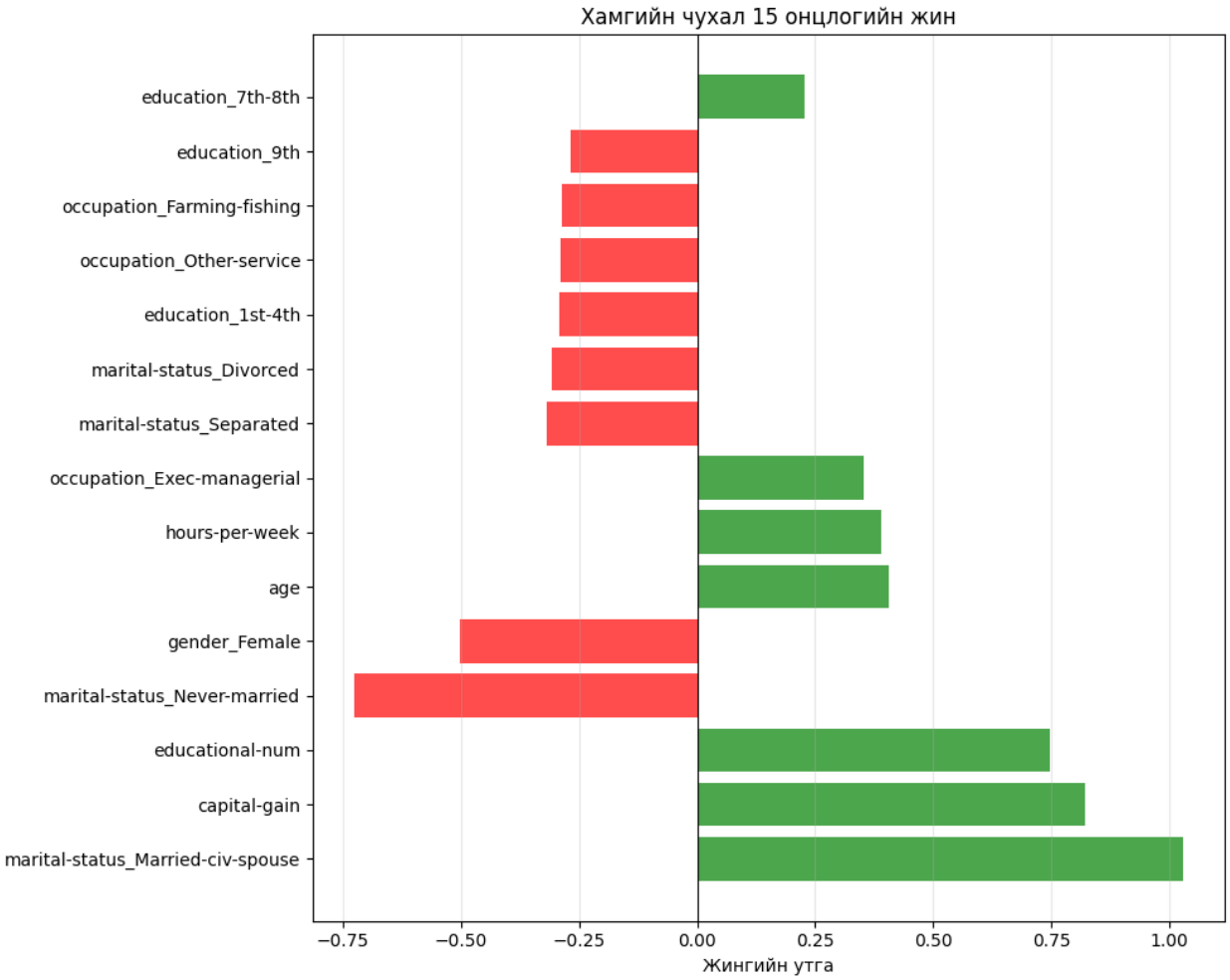
Зураг 2: Precision-Recall ба ROC муруйнууд

Зүүн талын Precision--Recall муруй нь бидний хамгийн том шийдвэрийн тэнцвэрийг илтгэнэ. Recall-ийг өсгөх үед илүү олон өндөр орлоготой хүмүүсийг илрүүлэх боломжтой ч, босго оноо доошлохын хэрээр Precision буурч, буруу эерэг таамаглал нэмэгддэг. Харин Precision-ийг сайжруулахын тулд илүү хатуу босго тавибал загвар зөв таамаглалд илүү итгэлтэй болох авч, олон жинхэнэ өндөр орлоготой хүмүүсийг орхигдуулах эрсдэлтэй. Энэ муруй нь хоёр классын үл тэнцвэртэй өгөгдлийн үед бодит гүйцэтгэлийг илүү үнэн зөв харуулдаг тул стратегийн хувьд түлхүүр үзүүлэлт болдог.

Баруун талын ROC муруй харьцангуй ерөнхий дүр зургийг өгч, манай загварын ялгах чадвар AUC = 0.891 гэдгийг харуулж байна. Энэ нь санамсаргүй таамаглалаас эрс илүү гүйцэтгэлтэй боловч ROC муруйн нь үл тэнцвэртэй өгөгдөлд хэт өгөөмөр ханддаг гэдгийг мартаж болохгүй. Иймээс өндөр AUC үзүүлэлт нь өөрөө хангалттай биш: практикт бид босго оноог бодитоор тохируулж, Precision ба Recall-ийн хооронд төслийн зорилгод нийцсэн зөв тэнцвэрийг олох шаардлагатай хэвээр байна.

4.7 Онцлогийн жин

Жингүүд нь онцлогийн чухал байдлыг харуулна. Эерэг жин = өндөр орлого руу түлхэх, сөрөг жин = бага орлого руу түлхэх:



Зураг 3: Хамгийн чухал 15 онцлогийн жин

Хамгийн өндөр нөлөөтэй хүчин зүйлс:

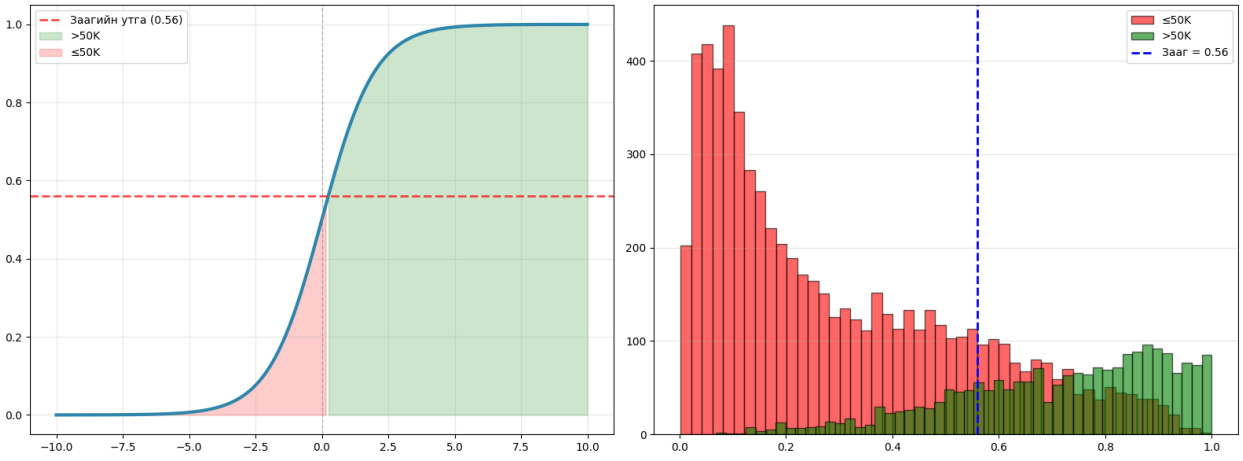
1. **educational-num (+0.89):** Боловсролын түвшин. Энэ нь хамгийн хүчтэй эерэг нөлөөтэй хүчин зүйл болж байна. Боловсрол өндөр байх тусам орлого өндөр байх магадлал эрс нэмэгдэнэ.
2. **capital-gain (+0.76):** Хөрөнгө оруулалтын ашиг. Санхүүгийн нэмэлт эх үүсвэртэй байх нь өндөр орлогын тод шинж тэмдэг юм.
3. **occupation_Other-service (-0.67):** Үйлчилгээний салбарын ажил. Энэ нь орлогод сөргөөр нөлөөлж байна, өөрөөр хэлбэл энэ салбарт ажиллагсад $\leq \quad 50K$ классд орох магадлал өндөр.
4. **marital-status_Never-married (-0.48):** Гэрлэж байгаагүй. Ганц бие хүмүүс гэр бүлтэй хүмүүстэй харьцуулахад орлого багатай байх хандлага ажиглагдсан.

5. **marital-status_Married-civ-spouse (+0.46):** Гэрлэсэн байдал. Эсрэгээрээ, гэр бүлийн тогтвортой байдал нь өндөр орлоготой эерэг хамааралтай байна.

Эдгээр үр дүн нь нийгэм, эдийн засгийн бодит байдалтай бүрэн нийцэж байна. Загвар маань зүгээр нэг тоо таах биш, бодит амьдралын зүй тогтлыг олж харсан гэж дүгнэж болно.

4.8 Сигмоид функц ба магадлалын тархалт

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 4: Сигмоид функц ба магадлалын тархалт

Гол ойлголт: 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.
- Энэ давхцал нь яагаад бид 100% нарийвчлалтай байж чадахгүйг тайлбарладаг. Зарим өндөр орлоготой хүмүүс бага орлоготой хүмүүстэй ижил шинж чанартай (эсвэл эсрэгээрээ) байгаа тул загвар тэднийг ялгахад хүндрэлтэй байна.

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, хэт тохируулалт

ажиглагдаагүй нь зөв регуляризацийн сонголт болон градиент бууруулалтын тохиргоо оновчтой байсны илрэл юм. Гэсэн хэдий ч гүйцэтгэл, ялангуяа accuracy нь 80% давахгүй байгаа нь өгөгдлийн бүтэц, классын тэнцвэргүй байдал зэрэгтэй холбоотой.

Ирээдүйд Random Forest, Gradient Boosting зэрэг илүү нарийн төвөгтэй загваруудыг ашиглавал >50K орлоготой хүмүүсийг илүү найдвартай таамаглах боломжтой. Гол сургамж нь зөвхөн Accuracy-аас гадна Precision, Recall, ROC/AUC зэрэг үзүүлэлтүүдийг ойлгон, загварын хязгаарлалт, өгөгдлийн чанарыг хамтад нь үнэлэх хэрэгтэй гэдгийг харуулж байна.

6 Багийн гишүүдийн оролцоо

7 Ишлэл зүүлт