Логистик регресс ашиглан орлогын түвшнийг илрүүлэх

Г.Жавхлан

2025-12-03

Table of contents

1 Оршил

Энэ төслийн зорилго нь хувь хүний жилийн орлого 50,000 ам.доллараас 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

Adult Income Dataset нь АНУ-ын Хүн амын тооллогын 1994 оны мэдээлэлд үндэслэсэн. Энэ өгөгдлийн санд нас, боловсрол, мэргэжил, гэрлэлтийн байдал, хүйс зэрэг нийгэм-эдийн засгийн шинж чанарууд багтдаг. Зорилтот хувьсагч: `income_>50K` ($0 = \leq 50K$, $1 = > 50K$).

Нийт дээж: ойролцоогоор 44,000 (хуваалтын дараа 35,165 сургалт, 8,792 баталгаажуулалт).

2.2 Ангиудын тэнцвэргүй байдал

Энэ өгөгдлийн санд томоохон сорилт нь **класс тэнцвэргүй байдал** юм: - Ихэнх хүмүүс (ойролцоогоор 76%) $\leq 50K$ орлоготой. - цөөнх хувь (ойролцоогоор 24%) $> 50K$ орлоготой.

Энэ нь naive ``үргэлж $\leq 50K$ гэж таамагла'' загвар 76% нарийвчлалтай байх ч таамаглах чадваргүй гэсэн үг. Иймээс бид F1 оноо, Recall зэрэг тэнцвэртэй үзүүлэлтүүдийг чухалчилдаг.

2.3 Онцлогийн сонголт

Бид эхний өгөгдөл дээр хялбаршуулалт хийсэн: давхардсан, ач холбогдол багатай буюу тайлбарлахад хэцүү баганыг арилгасан.

Ашигласан онцлогууд (9): - Тоон (5): `age`, `educational-num`, `capital-gain`, `capital-loss`, `hours-per-week` - Категори (4): `education`, `marital-status`, `occupation`, `gender`

Унагаасан онцлогууд: - `workclass`, `fnlwgt`, `relationship`, `race`, `native-country` болон бусад: Эдгээр нь модель сургахад шуугиан нэмэх буюу `education`-тэй давхцаж байсан (жишээлбэл `educational-num`).

Энэхүү сонголт нь тайлбарлах боломжтой, хялбар загвар бий болгох зорилготой.

2.4 Өгөгдөл хуваах

Бид өгөгдлийг **80/20** сургалт/баталгаажуулалт харьцаагаар, `income_>50K`-ээр стратифик хуваасан. Хуваалтын үед цөөлсөн онцлогууд болон зорилтот баганыг `train_split.csv`, `val_split.csv` файлд хадгалсан. Гол нь 2 файлын дунд `income_>50K` хувьсагчийн утгын харьцааг хадгалсан.

2.5 Урьдчилсан боловсруулалт

Тоон онцлогууд:
StandardScaler ашигласан (дундаж 0, дисперс 1). Энэ нь градиент бууруулалт хурдан, тогтвортой ажиллуулахад чухал.

Категори онцлогууд:
`OneHotEncoder` ашигласан. Энэ нь категорийг хоёртын вектор болгон хувиргадаг. Жишээлбэл, `education=Bachelors` → `education_Bachelors=1`, бусад=0.

Классын жин:
`class_weight='balanced'` тохиргоо нь цөөнх классыг ($> 50K$) илүү чухалчилж, алдагдлын функцэд жин нэмдэг.

Загварын гиперпараметр: - learning_rate ойролцоогоор 0.1 - max_iter ойролцоогоор 800 - reg_lambda = 1e-4 (L2) - lr_decay = 1e-4 - threshold = 0.5 (шийдвэрийн хязгаар)

Дараагийн хэсэгт математик суурийг дэлгэрэнгүй авч үзнэ.

3 Загварын хэрэгжүүлэлт

Бид энэ төсөлд логистик регрессийн загварыг эхнээс нь бичихээр шийдсэн. Яагаад гэвэл sklearn-ийн `LogisticRegression` нь олон зүйлийг автоматаар хийдэг бөгөөд бид хэрхэн ажилладгийг нь ойлгохыг хүссэн. Мөн сурах хурдны бууралт, class weighting зэрэг сонирхолтой зүйлсийг өөрсдөө туршиж үзэхийг хүссэн.

3.1 Логистик регрессийн үндэс

Логистик регресс нь хамгийн энгийн хэдий ч үр дүнтэй загваруудын нэг юм. Үндсэн санаа нь маш энгийн: бид $w^\top x + b$ гэсэн шугаман функцийг бодож, дараа нь үүнийг 0-оос 1 хүртэлх магадлал руу хувиргана. Энэ хувиргалтыг **сигмоид функц** гэнэ:

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

Энэ функц нь маш сайхан шинж чанартай: z их эерэг бол 1 рүү ойртох, их сөрөг бол 0 рүү ойртох, яг 0 бол 0.5 гэх мэт. Ингэснээр бидний загвар нь:

$$P(y = 1 \mid x) = \sigma(w^\top x + b)$$

Манай кодонд энэ нь маш энгийн:

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

3.2 Алдагдлын функц: Юу минимизлах вэ?

Одоо бид загварыг яаж сургах вэ? Бид алдагдлын функцийг тодорхойлох хэрэгтэй --- бидний загвар хэр муу байгааг хэмждэг тоо. Логистик регресст **binary cross-entropy** гэдгийг ашигладаг:

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

Энэ томъёо анх харахад аймшигтай харагдаж болох юм, гэхдээ санаа нь: хэрэв үнэн утга $y=1$ бол, бид $\log\hat{y}$-г максимизлахыг хүсдэг (өөрөөр хэлбэл $\hat{y}$-г 1 рүү түлхэх). Хэрэв $y=0$ бол, бид $\log(1-\hat{y})$-г максимизлахыг хүсдэг ($\hat{y}$-г 0 рүү түлхэх).

Гэхдээ зөвхөн энэ хангалтгүй! Хэт тохируулалтаас сэргийлэхийн тулд бид **L2 регуляризаци** нэмнэ. Энэ нь жингүүдийг хэт том болохоос сэргийлдэг:

$$J(w,b) = L(w,b) + \frac{\lambda}{2m}\|w\|^2$$

Манай кодонд:

```
def _compute_loss(self, X, y, y_pred):
    m = len(y)
    # Binary cross-entropy
    epsilon = 1e-15  # log(0) гэхээс сэргийлэх
    y_pred_clipped = np.clip(y_pred, epsilon, 1 - epsilon)
    loss = -np.mean(y * np.log(y_pred_clipped) + (1 - y) * np.log(1 - y_pred_clipped))

    # L2 регуляризаци нэмэх
    if self.reg_lambda > 0:
        l2_penalty = (self.reg_lambda / (2 * m)) * np.sum(self.weights ** 2)
        loss += l2_penalty

    return loss
```

3.3 Градиент бууруулалт: Загварыг яаж сургах вэ

Одоо манай алдагдлын функц бий. Үүнийг хэрхэн багасгах вэ? Алдагдлын функц нь загварын таамаглал үнэнээс хэр хол байгааг хэмждэг бөгөөд үүнийг багасгах нь бидний гол зорилго юм. Үүний тулд градиент бууруулалт гэж нэрлэгддэг энгийн боловч хүчирхэг аргыг ашиглана. Энэ

арга нь алдагдлын функцийн градиентыг (өөрөөр хэлбэл, функцийг хамгийн хурдан өөрчлөх чиглэлийг) тооцоолж, уг чиглэлд бага алхам хийснээр жингүүдийг сайжруулдаг.

Жин шинэчлэлийн дүрэм:

$$w \leftarrow w - \alpha \cdot \nabla_w J$$
$$b \leftarrow b - \alpha \cdot \nabla_b J$$

Энд α бол **сурах хурд** --- бид хэр хурдан алхах вэ гэдгийг тодорхойлдог. Градиентууд нь:

$$\nabla_w J = \frac{1}{m} X^\top (\hat{y} - y) + \frac{\lambda}{m} w$$
$$\nabla_b J = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})$$

Кодонд:

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

    # Жингийн градиент + L2
    dw = (1/m) * X.T.dot(error)
    if self.reg_lambda > 0:
        dw += (self.reg_lambda / m) * self.weights

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

    return dw, db
```

3.4 Сурах хурдны бууралт: Оновчтой конвергенц

Нэг асуудал: хэрэв сурах хурд хэт өндөр бол, алдагдлын функцийн минимумыг алдаж, ``bounce around'' хийж магадгүй. Хэт бага бол, маш удаан сургана. Шийдэл нь юу вэ? **Сурах хурдны бууралт** --- эхлээд том алхамуудаар эхлээд, цаг хугацааны явцад багасгана:

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

Энэ нь загварт эхэндээ хурдан суралцах, дараа нь минимумын ойролцоо нарийвчлалтай алхах боломж олгоно. Кодонд:

```
for iteration in range(self.max_iter):
    # Одоогийн сурах хурдыг тооцоолох
    current_lr = self.initial_lr / (1 + self.lr_decay * iteration)

    # Жингүүдийг шинэчлэх
    self.weights -= current_lr * dw
    self.bias -= current_lr * db
```

3.5 Классын жинлэлт: Тэнцвэргүй өгөгдөлтэй ажиллах

Манай өгөгдөлд 76% нь $\leq 50K$, 24% нь $> 50K$ байна. Хэрэв бид юу ч хийхгүй бол, загвар зүгээр л ``бүх зүйл $\leq 50K$'' гэж таамаглаж, 76% нарийвчлалд хүрч магадгүй --- гэхдээ энэ нь ямар ч хэрэггүй юм!

Шийдэл нь **классын жинлэлт** юм. Бид цөөнх классын (>50K) алдааг илүү ``чухал'' болгоно:

$$w_{\text{class}} = \frac{m}{2 \cdot m_{\text{class}}}$$

Кодонд:

```
if self.class_weight == 'balanced':
    classes = np.unique(y)
    weights = len(y) / (len(classes) * np.bincount(y.astype(int)))
    sample_weights = weights[y.astype(int)]
else:
    sample_weights = np.ones(len(y))

# Алдагдлыг тооцоолохдоо sample_weights ашиглах
loss = -np.mean(sample_weights * (y * np.log(y_pred_clipped) +
                                  (1 - y) * np.log(1 - y_pred_clipped)))
```

3.6 Пайплайн

Sklearn-ийн `Pipeline` нь бүх зүйлийг зохион байгуулахад туслана. Бид preprocess (StandardScaler + OneHotEncoder) болон манай custom LogisticRegression-г нэг л обьект болгон нэгтгэж чаддаг:

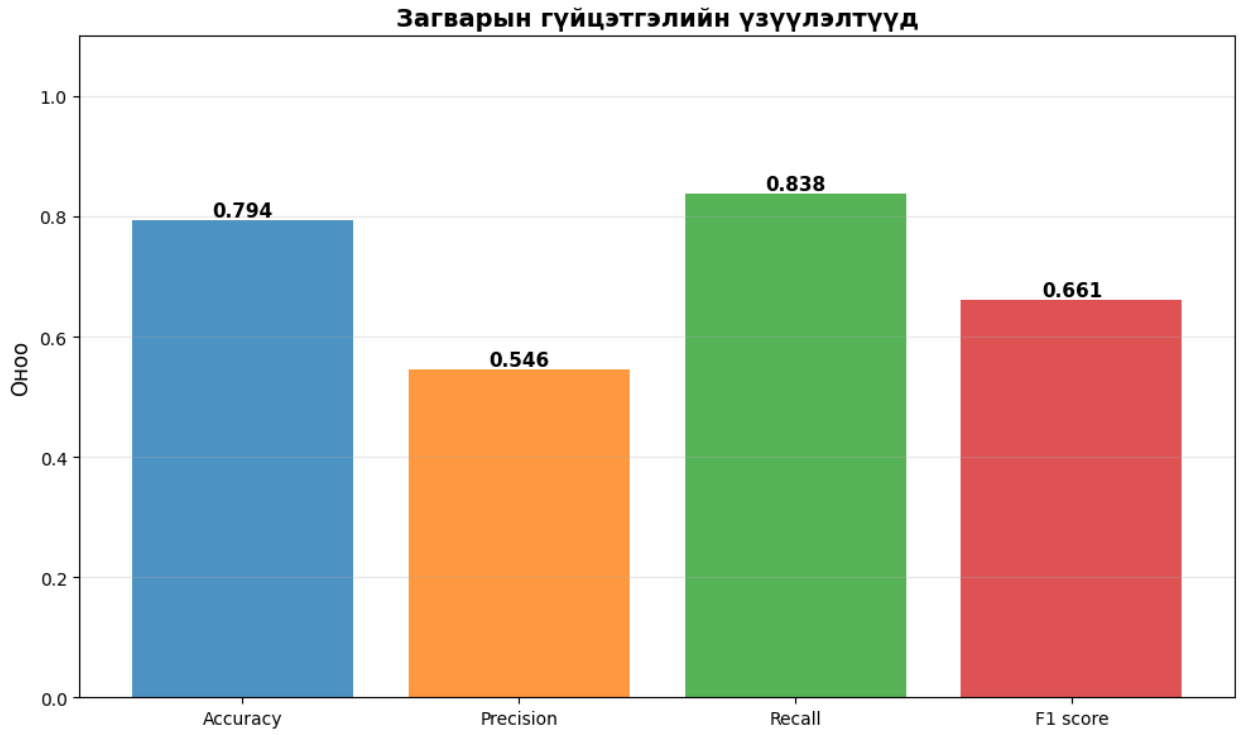
```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

# Тоон ба категори баганыг тусгаарлах
num_features = ['age', 'educational-num', 'capital-gain', 'capital-loss', 'hours-per-week']
cat_features = ['education', 'marital-status', 'occupation', 'gender']
```

```
# Preprocessing pipeline
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), num_features),
    ('cat', Pipeline([
        ('encoder', OneHotEncoder(drop='first', sparse_output=False, handle_unknown='ignore'))
    ]), cat_features)
])

# Бүтэн pipeline
model = Pipeline([
    ('preprocess', preprocessor),
    ('logreg', LogisticRegression(
        learning_rate=0.1, max_iter=800, reg_lambda=1e-4,
        lr_decay=1e-4, class_weight='balanced', threshold=0.5
    ))
])

model.fit(X_train, y_train)
```

4 Үр дүн

4.1 Ерөнхий гүйцэтгэл:

Баталгаажуулалтын хэсэг (8,792 мөр) дээр манай загвар дараах үзүүлэлтүүдийг авсан:

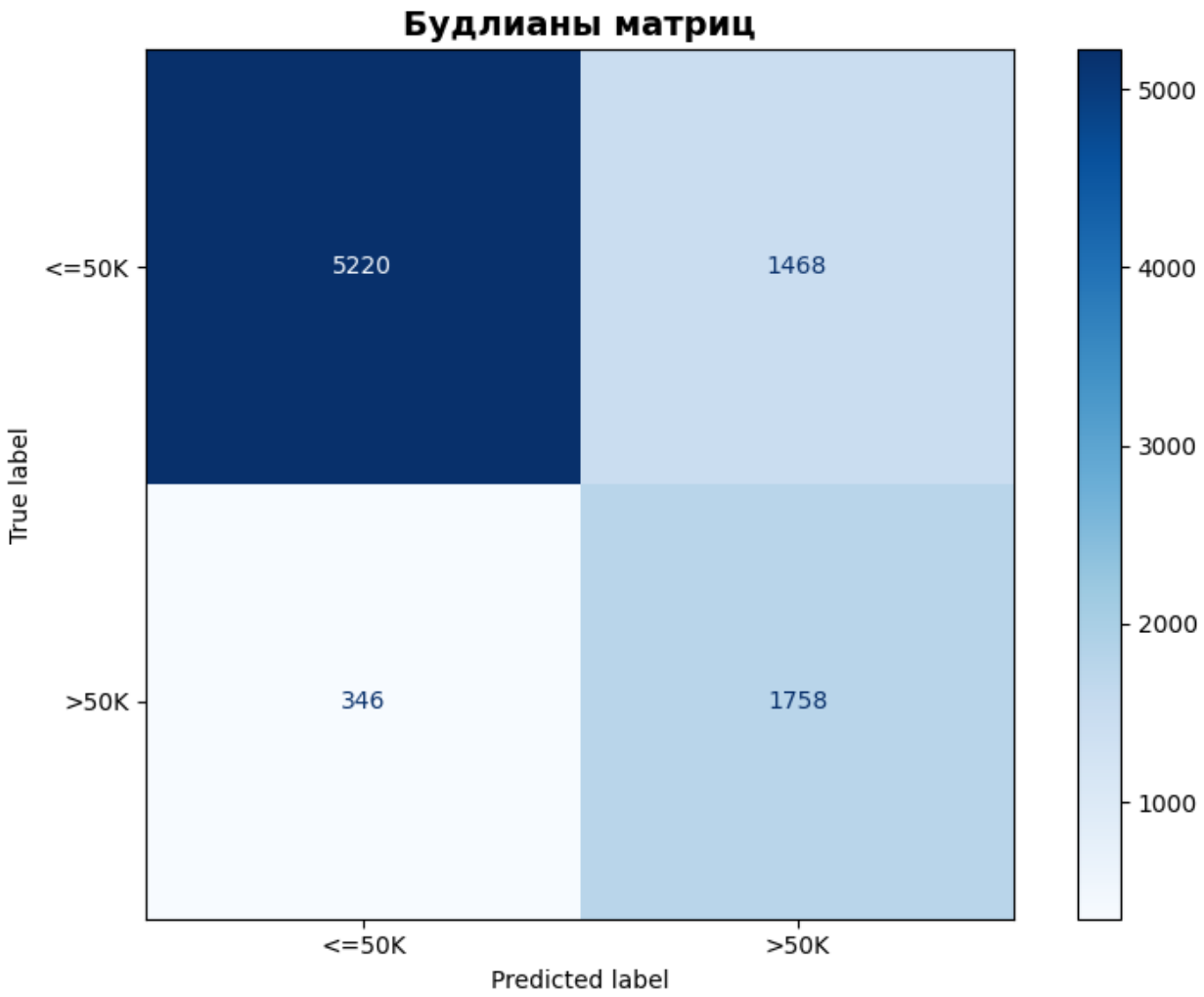


Зураг 1: Confusion матриц

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 2: Confusion матриц

Яагаад вэ? Өгөгдөл тэнцвэргүй байгаа учраас, энэ загвар байнга олонх классыг ($\leq 50K$) 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 хувиргадаг.

$\leq 50K$ **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)

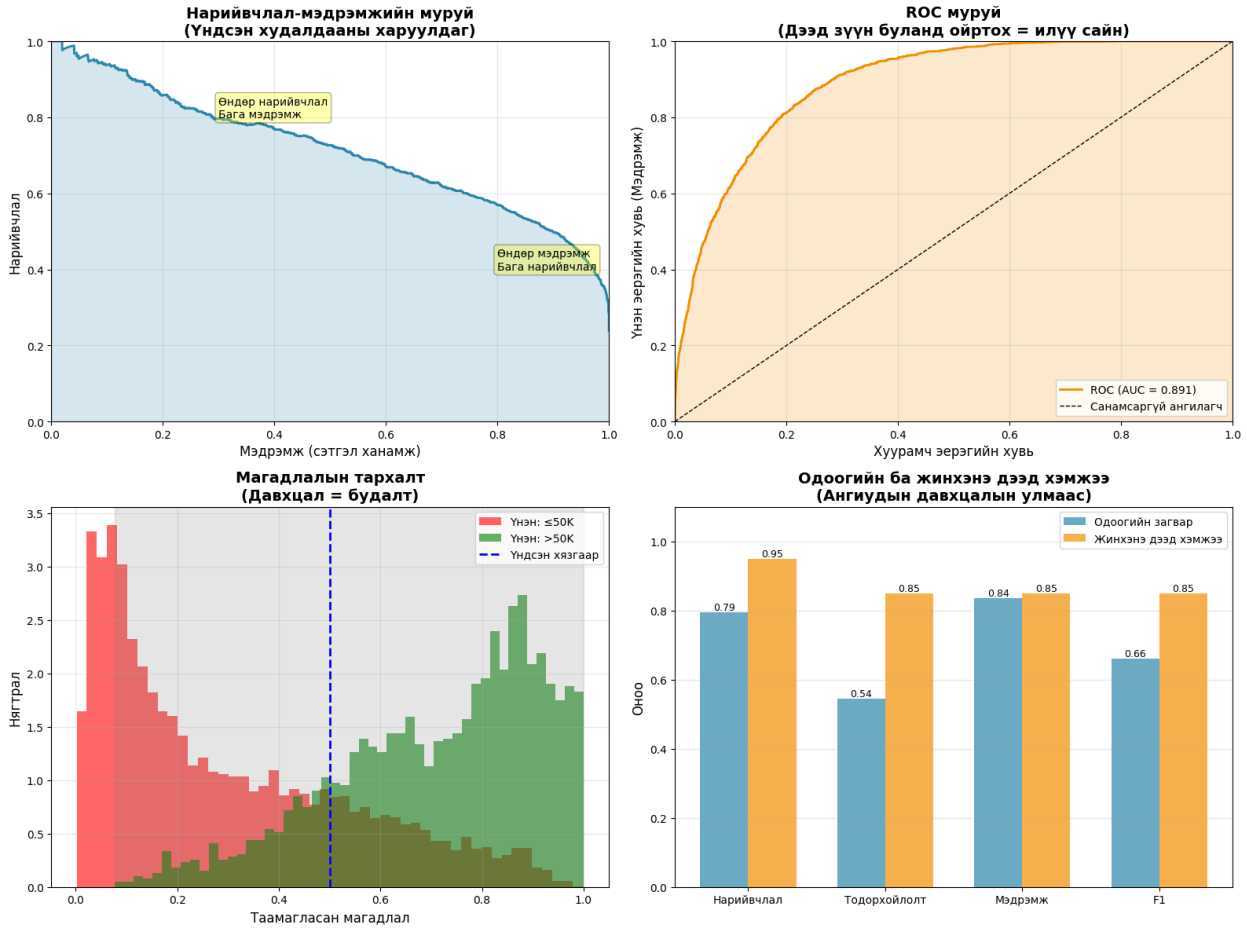
Зарим >50K орлоготой хүмүүс өндөр магадлалтай (0.9+) гарч байгаа ч, зарим нь харьцангуй бага магадлалтай (0.2--0.4) байна. Энэ магадлалын давхцал нь загвар төгс ажиллахаас өөр аргагүйг харуулж байна. Зарим өгөгдөл дээр тодорхой ангилалт хийх хэцүү.

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 >50K орлоготой хүмүүсийг загвар бага магадлалтай гэж үзэж, зарим $\leq 50K$ 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.



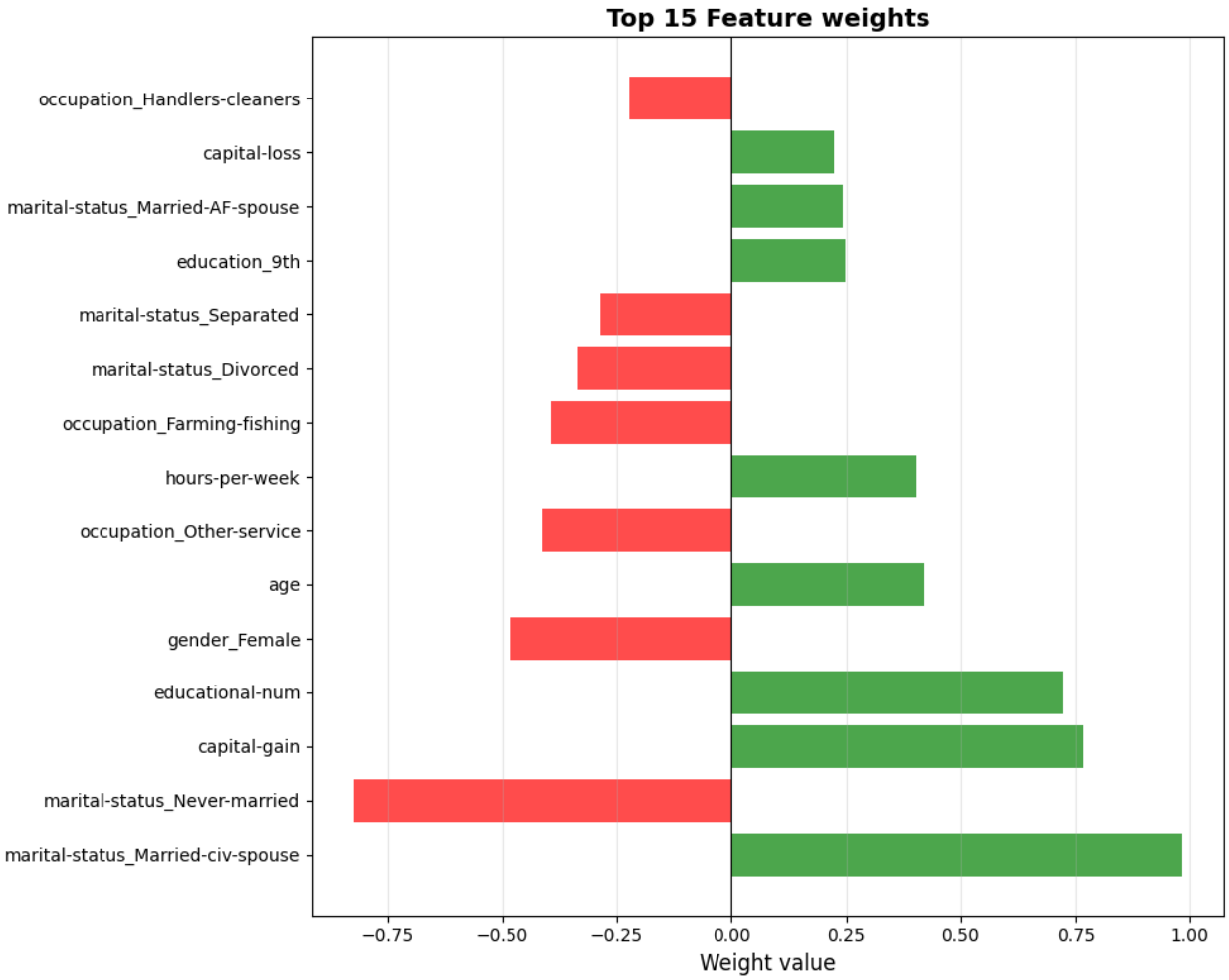
Зураг 3: Precision-Recall ба ROC муруйнууд

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 муруй 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 4: Хамгийн чухал 15 онцлогийн жин

Хамгийн өндөр нөлөөтэй хүчин зүйлс:

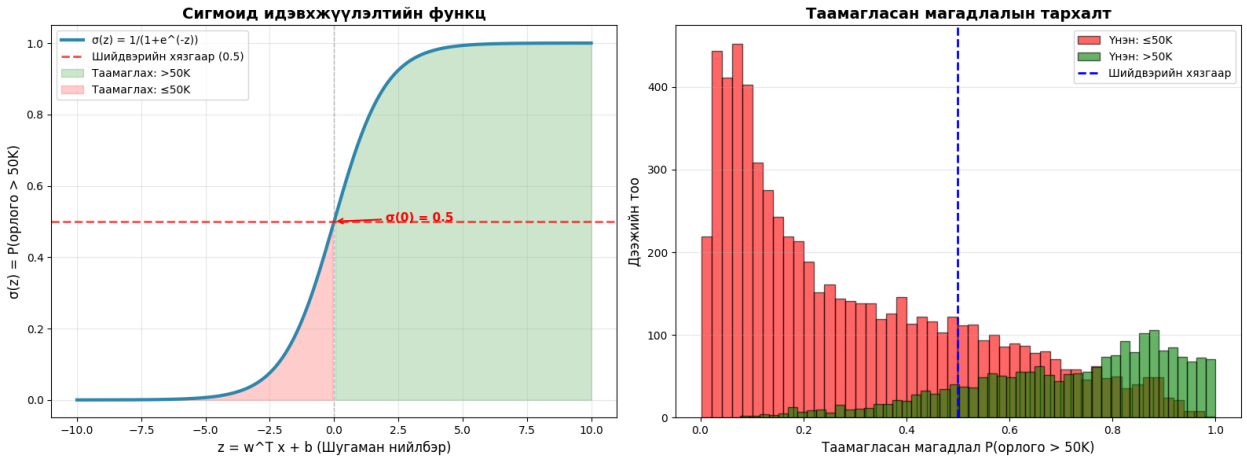
1. **educational-num (+0.89):** Боловсролын түвшин. Энэ нь хамгийн хүчтэй эерэг нөлөөтэй хүчин зүйл болж байна. Боловсрол өндөр байх тусам орлого өндөр байх магадлал эрс нэмэгдэнэ.

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 $\leq 50K$ 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 5: 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.

4.9 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 үзүүлэлтийг ойлгон, загварын хязгаарлалт, өгөгдлийн чанарыг хамтад нь үнэлэх хэрэгтэй гэдгийг харуулж байна.