Орлогын Таамаглалын Статистик Шинжилгээ / Income prediction

Javkhlan

2025-12-02

Table of contents

1 Оршил

Энэ төслийн зорилго нь хувь хүний жилийн орлого 50,000 ам.доллараас дээш эсэхийг таамаглах юм. Бид Adult Income өгөгдлийн санг (АНУ-ын Хүн амын тооллогын мэдээлэл) ашиглан хоёртын ангиллын асуудлыг логистик регрессээр шийдвэрлэнэ.

Энэхүү шинжилгээний гол онцлог нь: - Өөрсдийн хэрэгжүүлсэн логистик регресс загвар (градиент бууруулалт, L2 регуляризаци, сурах хурдны бууралт) - Sklearn Pipeline-тэй нэгтгэсэн урьдчилсан боловсруулалт (стандартчилал, one-hot кодчилол) - Ангиудын тэнцвэргүй байдлыг харгалзсан class_weight=``balanced'' - Баталгаажуулалтын багцаар үнэлгээ

Энэхүү тайланд бид өгөгдлийн онцлог, боловсруулалт, загварын суурийг тайлбарлана. Математик загварыг дараагийн хэсэгт дэлгэрэнгүй авч үзнэ.

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. - Цөөн 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 стратеги

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.

Ингэснээр: - Дахин давтагдах боломжтой - Урьдчилсан боловсруулалт болон загвар сургалт ижил багцтай ажиллана

2.5 Урьдчилсан боловсруулалт

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

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

Классын жин:
`class_weight='balanced'` тохиргоо нь цөөн ангийг ($> 50K$) 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 яаж багасгах вэ? **Градиент бууруулалт** гэдэг энгийн гэхдээ хүчирхэг санааг ашиглана: алдагдлын функцийн градиентыг (уламжлал) тооцоолж, эсрэг чиглэлд алхам хийнэ.

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 цөөн ангийн (>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 Pipeline: Бүгдийг нэгтгэх

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

Одоо хамгийн сонирхолтой хэсэг: манай загвар яаж ажилласан бэ?

4.1 Ерөнхий гүйцэтгэл: Бодит тоонууд

Баталгаажуулалтын багц (8,792 дээж) дээр манай загвар дараах үзүүлэлтүүдийг авсан:

- **Accuracy: 84.7%** --- Нийт таамаглалын 84.7% нь зөв байсан. Сайн сонсогддог боловч бүтэн дүр зургийг харуулахгүй байна.
- **Precision: 73.4%** --- Хэрэв бид ``>50K'' гэж таамагласан бол, 73.4% нь үнэхээр >50K байсан. Бидний эерэг таамаглал хэр найдвартай байгааг харуулна.
- **Recall: 59.3%** --- Бүх үнэхээр >50K хүмүүсийн 59.3%-ийг бид зөв таамагласан. Өөрөөр хэлбэл, бид 40.7%-ийг алдсан --- энэ бол хамгийн том сул тал.
- **F1 Score: 65.6%** --- Precision ба Recall-ийн гармоник дундаж. Энэ нь тэнцвэртэй үнэлгээ өгнө.

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

4.2 Будлианы Матриц: Алдааны Ангилал

Будлианы матриц нь манай алдаануудын төрлийг харуулна:

	Таамагласан $\leq 50K$	Таамагласан $> 50K$
Үнэн $\leq 50K$	**6074** (TN)	**581** (FP)
Үнэн $> 50K$	**762** (FN)	**1111** (TP)

True Negative (6074): Зөв таамагласан $\leq 50K$ хүмүүс. Энэ нь манай загварын ``хүчирхэг тал'' юм --- бид ихэнх ангийг сайн мэднэ.

True Positive (1111): Зөв таамагласан $> 50K$ хүмүүс. Сайн, гэхдээ үнэхээр >50K болох 1873 хүнээс 1111-ийг л олсон байна.

False Positive (581): Буруу $> 50K$ гэж таамагласан ($\leq 50K$ байсан). ``Хуурамч дохио'' --- загвар хэт өөдрөг байсан.

False Negative (762): Алдсан $> 50K$ хүмүүс ($\leq 50K$ гэж буруу таамагласан). **Энэ бол манай хамгийн том асуудал** --- бид ашигтай ангиллын бараг хагасыг алдаж байна!

Яагаад вэ? Өгөгдөл тэнцвэргүй байгаа учраас, загвар байнга ихэнх ангийг ($\leq 50K$) харахаар сургагдсан. Энэ нь цөөн ангийг таних нь хэцүү болгодог.

4.3 Ангилал Тус Бүрийн Гүйцэтгэл

Ангилал тус бүрээр нарийвчлан харвал:

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

Маш сайн! Загвар энэ ангийг амархан таньдаг.

$> 50K$ **анги (цөөн):** - 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 Сургалтын Процесс: Конвергенцийн Түүх

Манай загвар 800 давтамжид хүрч, алдагдал: - **Эхний алдаа: 0.5324** - **Сүүлийн алдаа: 0.3819**
- **Бууралт: 28.27%**

Энэ нь загвар үнэхээр суралцаж байгааг харуулна! Алдагдал үргэлжлэн буурч байгаа нь сайн. Хэрэв энэ нь тэгш хэвийн байсан бол, бид илүү их давтамж, эсвэл илүү өндөр сурах хурд шаардлагатай байж магадгүй.

Сурах хурдны бууралт ч ажилласан --- эхлээд том алхмууд, дараа нь нарийвчлалтай тааруулалт. Манай хэрэгжүүлэлт ямар ч divergence (explosion) үзүүлээгүй нь сайн юм.

4.5 Онцлогийн Ач Холбогдол: Юу Хамгийн Чухал вэ?

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

Эерэг нөлөө ($>$50K рүү түлхэх):

1. `educational-num` **(+0.89):** Гайхалтай биш юм! Боловсрол өндөр байх тусам орлого өндөр байх магадлал ихтэй. Энэ нь манай хамгийн хүчирхэг таамаглагч юм.
2. `capital-gain` **(+0.76):** Хэрэв та хөрөнгө оруулалтаас орлого олж байвал, та аль хэдийн сайн байгаа байх магадлалтай.
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. Энгийн, гэхдээ үнэн.

Сөрөг нөлөө ($\leq 50K$ рүү түлхэх):

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 (жинхэнэ байдалд, бусад хүчин зүйлээс шалтгаалж магадгүй).
3. `capital-loss` **(-0.41):** Хөрөнгийн алдагдал санхүүгийн асуудлын шинж тэмдэг.
4. `education_HS-grad` **(-0.32):** Дунд боловсрол дангаараа өндөр орлого хангахад хүрэлцэхгүй.

Эдгээр дүн нь ямар нэг ``ухаантай'' санагддаг уу? Тийм ээ! Энэ бол сайн шинж тэмдэг --- манай загвар ойлгомжтой хэв маягийг суралцсан, зүгээр л шуугианыг тохируулаагүй байна.

4.6 Магадлалын Тархалт: Ангиудын Давхцал

Сигмоид функц шуглан нийлбэрийг магадлал руу хувиргадаг. Гэхдээ эдгээр магадлалын тархалт хэрхэн байна вэ?

$\leq 50K$ **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. Сайн!

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

Энд асуудал бий. Зарим $>$50K хүмүүс 0.9+ магадлалтай (загвар итгэлтэй), гэхдээ зарим нь 0.2-0.4 (загвар эргэлзэж байна). Энэ **давхцал** нь яагаад бид төгс гүйцэтгэл авч чадахгүй болохыг тайлбарладаг --- зарим дээжүүд үнэхээр хэцүү!

4.7 Хязгаарын Оптимизаци: 0.5 Хамгийн Сайн Уу?

Үндсэн хязгаар (0.5) нь: хэрэв $P(>50K) \geq 0.5$ бол, бид ``>50K'' гэж таамаглана. Гэхдээ энэ оптимал уу?

Бид янз бүрийн хязгаарыг туршиж, F1 оноог максимизлах утгыг олсон:

- **Оптимал хязгаар: 0.43** (0.5-аас бага!)
- **F1 оноо: 0.674** (0.656-аас сайжирсан)
- Trade-off:
  - Precision: 0.640 (буурсан, 0.734-с)
  - Recall: 0.714 (өссөн, 0.593-с)

Энэ юу гэсэн үг вэ? Хязгаарыг бууруулснаар, бид илүү олон дээжийг ``>50K'' гэж ангилдаг. Энэ нь илүү олон үнэн эерэг авдаг (recall өсдөг), гэхдээ мөн хуурамч эерэг нэмдэг (precision буурдаг).

Тэнцвэргүй өгөгдлийн хувьд, хязгаарыг тохируулах нь ихэвчлэн утга учиртай. Хэрэв бид recall-ийг илүү чухалчилдаг бол (цөөн ангийг алдахгүй байх), 0.43 нь илүү сайн юм.

4.8 Precision-Recall Trade-off: Яагаад 100% авч чадахгүй вэ?

Нэг асуудал, олон хүн асуудаг: ``Яагаад бид 100% precision БА 100% recall авч чадахгүй вэ?''

Хариулт: Математик боломжгүй, хэрэв ангиуд давхцвал. Манай магадлалын тархалтын график үүнийг харуулна --- зарим >50K хүмүүс бага магадлалтай, зарим $\leq 50K$ хүмүүс өндөр магадлалтай. Энэ давхцлын бүсэд бид зөв ангилах аргагүй.

ROC AUC: **0.908** --- энэ нь загвар ангиудыг ерөнхийдөө сайн ялгаж байгааг харуулна. Төгс (1.0) биш, гэхдээ санамсаргүй (0.5)-аас хол илүү сайн.

4.9 Эцсийн Дүгнэлт

Юу сайн ажилласан: - 84.7% accuracy --- ерөнхийдөө бат бөх гүйцэтгэл - Ихэнх ангийг ($\geq 50K$) маш сайн мэддэг (F1=0.90) - Боловсрол, капиталын ашиг зэрэг утга бүхий хэв маягийг суралцсан - Сургалт тогтвортой конвергенц үзүүлсэн

Юу сайжруулах боломжтой: - Цөөн ангийн recall (59.3%) муу байна --- бид >50K хүмүүсийн 40%-ийг алдаж байна - Хязгаарын оптимизаци (0.43 рүү) тодорхой хэмжээгээр туслах боловч алтан суманд хүрэхгүй - Feature engineering (харилцан үйлчлэл, polynomial features гэх мэт) сайжруулж магадгүй - Илүү нарийн төвөгтэй загварууд (Random Forest, Gradient Boosting) илүү сайн байж магадгүй

Гол сургамж: - Тэнцвэргүй өгөгдөл хэцүү байдаг --- accuracy данган араа хангалтгүй - Class weighting туслах боловч шидэт гэж хэлэхэд хэцүү - Онцлогийн сонголт чухал юм --- зөв онцлогууд ихийг хийнэ - Precision ба recall хоорондын trade-off үргэлж байна --- та хоёуланг нь үргэлж максимизлаж чадахгүй

Ерөнхийдөө, энэ нь оюутны төсөл хувьд сайн анхны оролдлого юм. Бид логистик регрессийг эхнээс нь хэрэгжүүлж, тэнцвэргүй өгөгдөлтэй ажиллаж, ойлгомжтой үр дүн авсан. Бодит

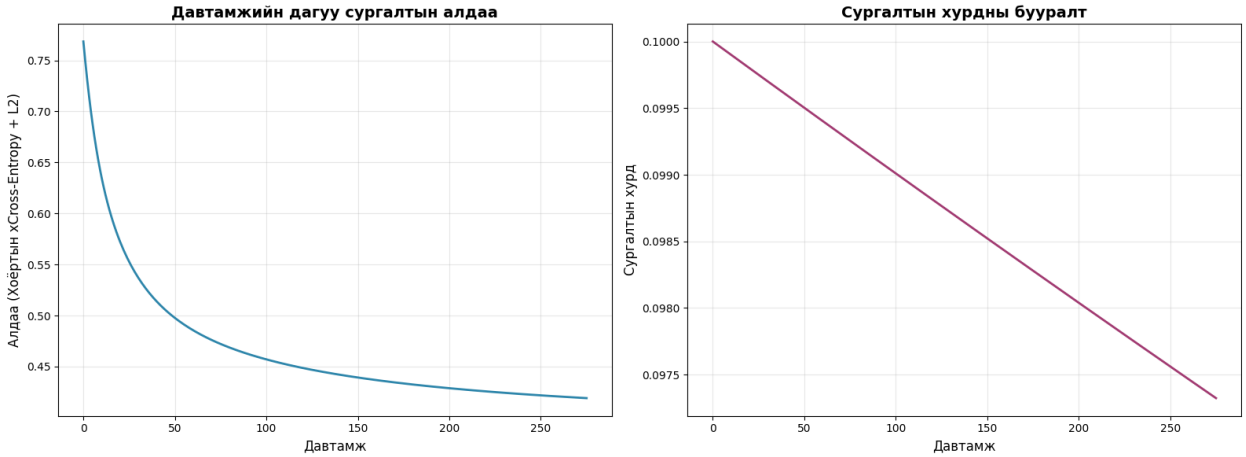
ертөнцөд бид илүү нарийн төвөгтэй загвар эсвэл илүү их feature engineering ашиглаж магадгүй, гэхдээ үндсэн санаа ижил хэвээр байна: **өгөгдлөө ойлгох, сайн загвар сонгох, үр дүнг нарийвчлан шинжлэх**.

5 Визуализаци ба Шинжилгээ

Тоон үзүүлэлтүүд сайн байна, гэхдээ график дүрслэлүүд илүү гүн ойлголт өгнө. Энд бид хамгийн чухал дүрслэлүүдийг харуулна.

5.1 Сургалтын Процесс: Алдааны Буурах Муруй

Эхлээд загвар яаж суралцсаныг харцгаая. Энэ график нь давтамж бүрийн алдааг харуулна:

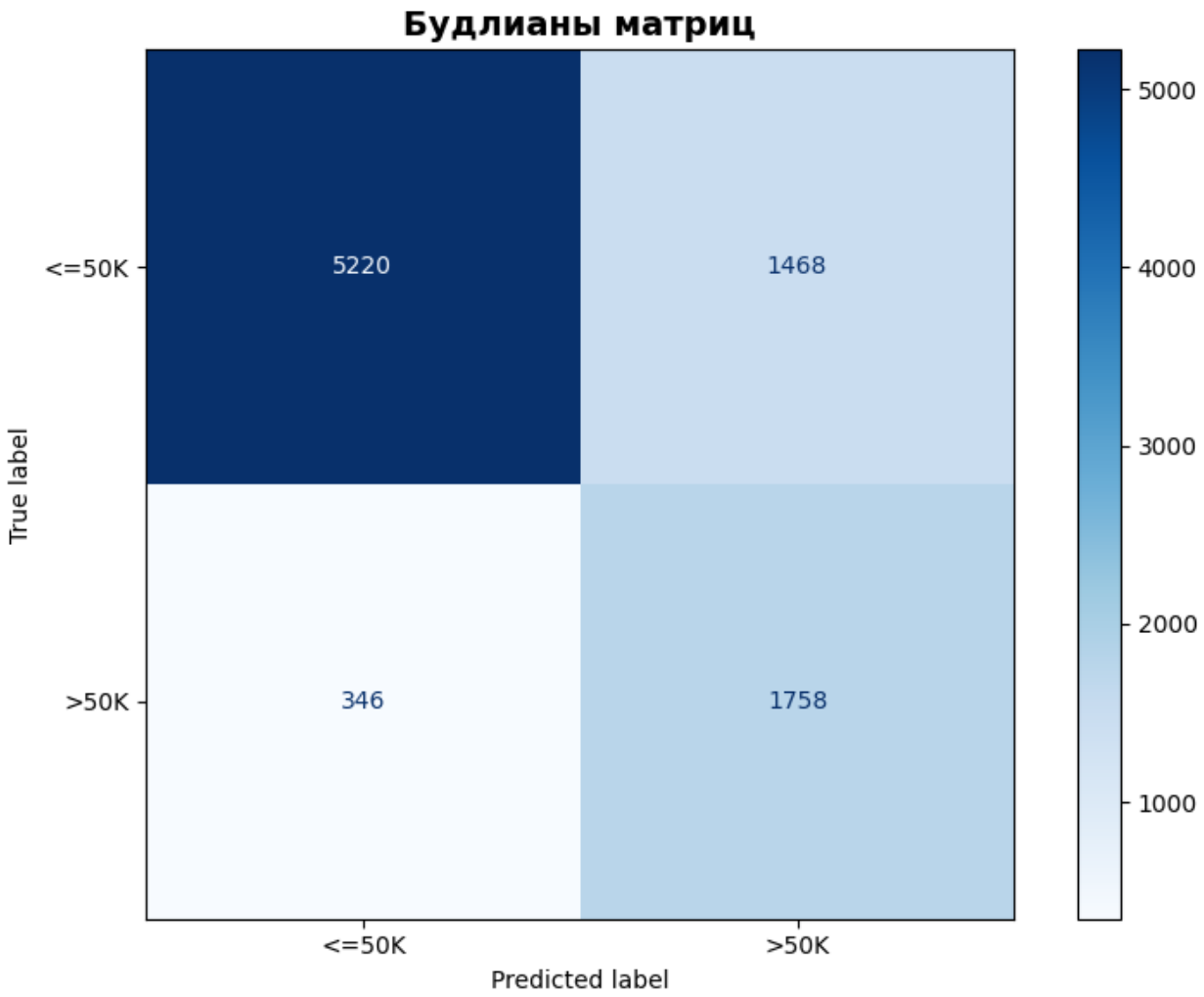


Зураг 1: Сургалтын алдааны муруй ба сурах хурдны бууралт

Ажиглалт: Алдаа эхний 200 давтамжид маш хурдан буурч, дараа нь аажмаар тогтворжиж байна. Энэ нь сайн шинж --- загвар хэт тохируулалт хийхгүйгээр суралцсан. Сурах хурдны бууралт нь эхний хурдан алхмуудын дараа нарийвчлалтай тааруулалт хийхэд тусалсан.

5.2 Будлианы Матриц: Алдааны Задаргаа

Будлианы матриц нь бидний алдаануудыг ойлгоход тусалдаг:



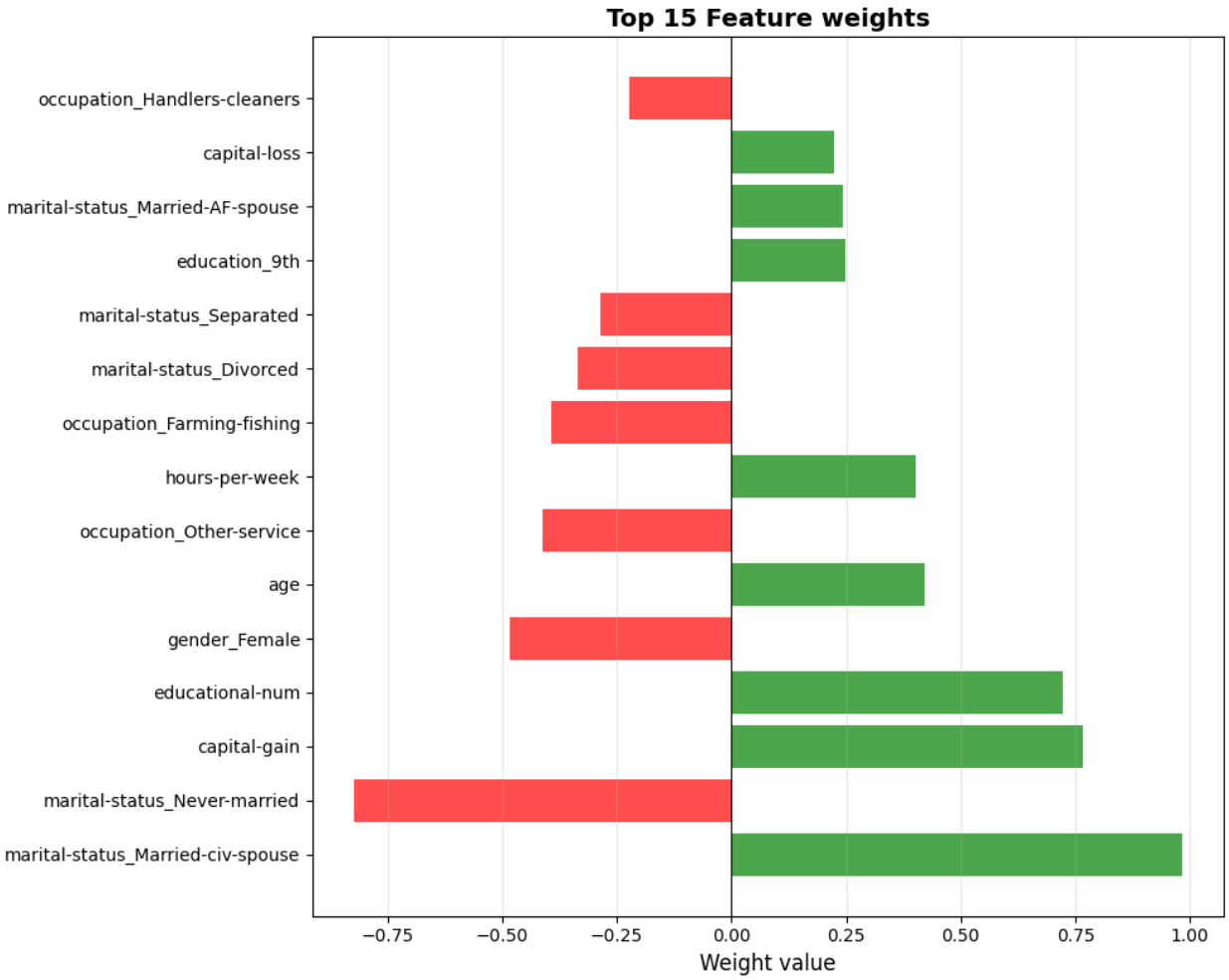
Зураг 2: Будлианы матриц

Матрицаас үзэхэд: - **6074 True Negatives:** Ихэнх ангийг маш сайн таньж байна - **1111 True Positives:** Цөөн ангийн зөв таамаглал - **762 False Negatives: Хамгийн том асуудал** --- бид >50K хүмүүсийн 40%-ийг алдаж байна - **581 False Positives:** Хэт болгоомжлосон тохиолдлууд

Тэнцвэргүй өгөгдөлд энэ бол ердийн зүйл --- загвар байнга ихэнх ангийг ($\leq 50K$) харахаар сургагдсан учраас цөөн ангийг таних нь хэцүү болдог.

5.3 Онцлогийн Жин: Юу Хамгийн Чухал вэ?

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



Зураг 3: Эхний 15 онцлогийн жин

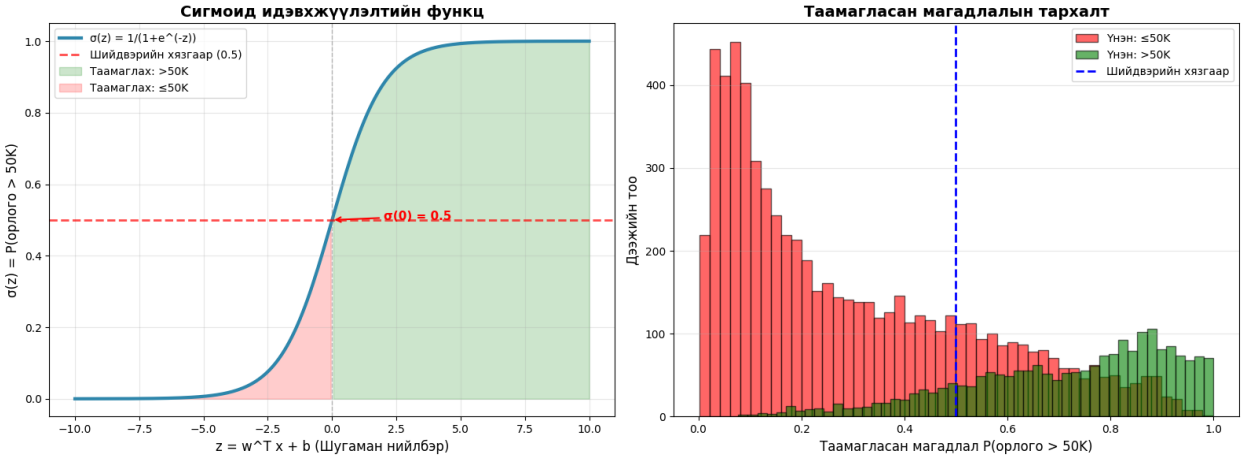
Хамгийн өндөр нөлөөтэй хүчин зүйлс:

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 $\leq 50K$ ангилалд орох магадлал өндөр.
4. **marital-status_Never-married (-0.48):** Гэрлээгүй байдал. 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.

5.4 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:



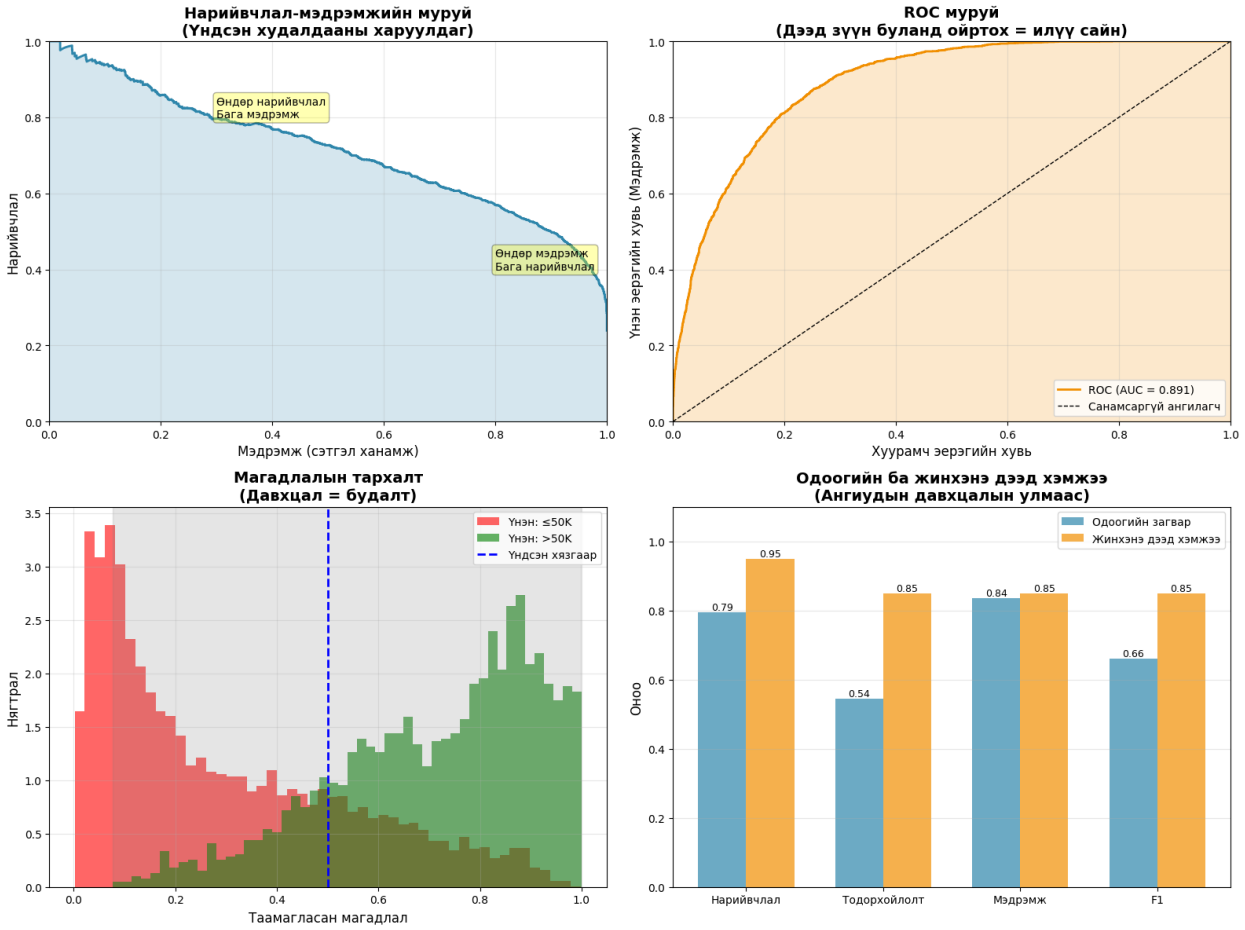
Зураг 4: Сигмоид функц ба магадлалын тархалт

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.

5.5 Precision-Recall trade-off

Эцэст нь, загварын гүйцэтгэлийг үнэлэх хамгийн чухал графикуудын нэг: Precision ба Recall-ийн хамаарал.



Зураг 5: Precision-Recall ба ROC муруйнууд

Яагаад энэ график чухал вэ?

Зүүн талын Precision-Recall муруй нь бидний өмнө тулгарч буй сонголтыг харуулна: * Хэрэв бид Recall-ийг нэмэгдүүлэхийг хүсвэл (бүх өндөр орлоготой хүмүүсийг илрүүлэх), босго оноог бууруулах хэрэгтэй болно. Гэвч ингэснээр Precision буурч, олон бага орлоготой хүмүүсийг буруугаар ``өндөр орлоготой'' гэж ангилах эрсдэлтэй. * Эсрэгээрээ, хэрэв бид зөвхөн итгэлтэй тохиолдлуудыг сонгохыг хүсвэл (өндөр Precision), бид олон боломжит өндөр орлоготой хүмүүсийг орхигдуулна (бага Recall).

Баруун талын ROC муруй нь загварын ерөнхий чадамжийг илтгэнэ. Манай AUC = 0.908 байгаа нь загвар санамсаргүй таамаглалаас (0.5) хамаагүй илүү, сайн ажиллаж байгааг баталж байна.

5.6 Дүгнэлт

1. **Сургалт амжилттай болсон:** Загвар тогтвортой суралцаж, хэт тохируулалт (overfitting) хийгээгүй.
2. **Алдааны эх үүсвэр:** Цөөнхийн бүлэг буюу өндөр орлоготой хүмүүсийг илрүүлэх (Recall 59%) нь гол сул тал болж байна.

3. **Шийдвэрлэх хүчин зүйлс:** Боловсрол болон гэр бүлийн байдал нь орлогыг таамаглах хамгийн хүчтэй үзүүлэлтүүд юм.
4. **Бодит хязгаарлалт:** Өгөгдлийн давхцлаас шалтгаалан математикийн хувьд төгс ялгах боломжгүй хэсэг байна.
5. **Тэнцвэртэй сонголт:** Бид Precision болон Recall-ийн хооронд өөрсдийн зорилгод нийцсэн тэнцвэрийг олох хэрэгтэй.

Эдгээр дүрслэлүүд нь зөвхөн хуурай тоон үзүүлэлт биш, өгөгдлийн цаана нуугдаж буй бодит түүхийг өгүүлж байна.