

Цителадзе Г.А. 5030102/20401

**Задание:** научиться предсказывать, уйдет ли сотрудник из компании. Модель - логистическая регрессия. Нужно сделать EDA, обучить модель, подобрать гиперпараметры.

**Репозиторий с кодом:** [https://github.com/tsitic/ML\\_y4](https://github.com/tsitic/ML_y4)

## 1. EDA

Для проведения первичного разведочного анализа данных был написан отдельный скрипт `first_anlz.py`, который выполняет группировку признаков по целевой переменной `Attrition` и выводит основные статистические характеристики (`count`, `mean`, `std`, квартили) для числовых признаков, а также доли классов для категориальных признаков.

В результате анализа было установлено следующее:

1) Целевая переменная сильно несбалансирована: доля сотрудников, покинувших компанию, составляет около 12%, что указывает на необходимость учитывать дисбаланс классов при обучении модели.

2) Среди **числовых признаков** наибольшее влияние на уход сотрудников показали:

`MonthlyIncome` - ушедшие сотрудники в среднем имеют заметно более низкий доход

`Age` - более молодые сотрудники чаще покидают компанию

`JobLevel` - сотрудники с низким уровнем должности уходят чаще

`StockOptionLevel` - наличие опционов связано с большей лояльностью признаки стажа (`TotalWorkingYears`, `YearsAtCompany`, `YearsInCurrentRole`, `YearsWithCurrManager`) - у ушедших сотрудников значения в среднем ниже

Ряд числовых признаков показал слабую или отсутствующую связь с целевой переменной, в частности:

`PerformanceRating`, `StandardHours`, `EmployeeCount`, `Gender`, а также такие признаки, как `MonthlyRate`, `HourlyRate`, `TrainingTimesLastYear`, `WorkLifeBalance`.

3) Среди **категориальных** признаков были выявлены категории с повышенным риском ухода:

`OverTime = Yes` - один из наиболее сильных факторов ухода;

BusinessTravel = Travel\_Frequently - сотрудники с частыми командировками уходят заметно чаще;

JobRole = Sales Representative, Human Resources, Laboratory Technician - наиболее рискованные должности;

MaritalStatus = Single - доля ушедших сотрудников значительно выше по сравнению с другими категориями.

На основе проведённого разведочного анализа данных было принято решение отказаться от полного one-hot кодирования всех категориальных признаков и вместо этого:

удалить константные и маловлияльные признаки

сохранить только наиболее информативные числовые признаки

вынести наиболее рискованные категории в отдельные бинарные признаки

использовать упорядоченное кодирование для признаков, имеющих естественный порядок (BusinessTravel, MaritalStatus).

## 2.Подготовка данных

Был написан файл processor.py в нем было сделано следующее:

Удалены константы и неинформативные признаки, которые точно не потребуются модели:

```
drop_cols = [
    "id",           #индекс
    "EmployeeCount", #всегда 1
    "Over18",        #всегда Y
    "StandardHours", #всегда 80
    "Gender",        #почти не влияет
    "PerformanceRating", #почти константа
]
df = df.drop(columns=[c for c in drop_cols if c in df.columns])
```

Выделена целевая переменная Attrition в отдельный столбец

```
y = df[TARGET].astype(int)
df = df.drop(columns=[TARGET])
```

Был сформирован список числовых признаков, показавших наибольшую связь с уходом сотрудников (менялся в процессе работы):

```
keep_num = [
    "Age",
    "MonthlyIncome",
    "JobLevel",
    "StockOptionLevel",
    "TotalWorkingYears",
    "YearsAtCompany",
    "YearsInCurrentRole",
    "YearsWithCurrManager",
    "EnvironmentSatisfaction",
    "JobInvolvement",
    "RelationshipSatisfaction",
    "JobSatisfaction",
    "DistanceFromHome",
    "PercentSalaryHike",
    "NumCompaniesWorked",
]
```

Для признака `MonthlyIncome`, в EDA была замечена асимметрия распределения, для него применено логарифмическое преобразование:

```
df_num["MonthlyIncome_log"] = np.log1p(df_num["MonthlyIncome"])
df_num = df_num.drop(columns=["MonthlyIncome"])
```

И произведено кодирование категориальных признаков  
Переработки:

```
df_num["OverTime_bin"] = (df["OverTime"] == "Yes").astype(int)
```

Частота командировок:

```
travel_map = {
    "Non-Travel": 0.0,
    "Travel_Rarely": 0.5,
    "Travel_Frequently": 1.0
}
df_num["BusinessTravel_code"] = df["BusinessTravel"].map(travel_map).fillna(0.0)
```

Семейное положение:

```
marital_map = {
    "Divorced": 0.3,
    "Married": 0.5,
    "Single": 1.0
}
df_num["MaritalStatus_code"] = df["MaritalStatus"].map(marital_map).fillna(0.5)
```

Рискованные категории должностей и отделов:

```
df_num["is_SalesRepresentative"] = (df["JobRole"] == "Sales Representative").astype(int)
df_num["is_JobRole_HR"] = (df["JobRole"] == "Human Resources").astype(int)
df_num["is_LaboratoryTechnician"] = (df["JobRole"] == "Laboratory Technician").astype(int)
```

Также были добавлены производные признаки, так была добавлена доля стажа в текущей компании от общего стажа, 5 лет в компании при общем стаже 6 лет – признак стабильности, а 5 лет в компании при общем стаже 30 лет – это совершенно другая ситуация.

```
df_num["years_in_company_share"] = df_num["YearsAtCompany"] / (df_num["TotalWorkingYears"] + 1.0)
```

Аналогично была добавлена доля времени с текущим руководством:

```
df_num["years_with_manager_share"] = df_num["YearsWithCurrManager"] / (df_num["YearsAtCompany"] + 1.0)
```

В конце processor.py формируем финальный датасет:

```
df_out[TARGET] = y.values

df_out = df_out.fillna(0)

df_out.to_csv("data_f.csv", index=False)
```

### 3.Логистическая регрессия

Была реализована модель логистической регрессия (файл logistic\_model.py).

Заданы основные гиперпараметры  
LEARNING\_RATE –шаг градиентного спуска  
EPOCHS -число эпох обучения  
L2\_REG – коэффициент для L2 регуляризации  
TEST\_SIZE, VAL\_SIZE -доля тестовой и валидационной выборки.  
POS\_WEIGHT – для учёта дисбаланса классов  
SEEDS – набор значений для оценки модели на разных seeds.

Реализована логистическая функция и bias:

```
def sigmoid(z):
    z = np.clip(z, -500, 500)
    return 1.0 / (1.0 + np.exp(-z))

def add_bias(X):
    return np.hstack([np.ones(X.shape[0], 1), X])
```

Реализована функция стандартизации признаков:

```
def standardize(train, val, test):
    add mean = train.mean(axis=0)
    std = train.std(axis=0)
    std[std == 0] = 1
    return (train - mean) / std, (val - mean) / std, (test - mean) / std
```

И реализовано разбиение выборки:

```
def stratified_split_indices(y, test_size, seed):
    rng = np.random.default_rng(seed)
    idx0 = np.where(y == 0)[0]
    idx1 = np.where(y == 1)[0]
    rng.shuffle(idx0)
    rng.shuffle(idx1)

    n0_test = int(len(idx0) * test_size)
    n1_test = int(len(idx1) * test_size)

    test_idx = np.concatenate([idx0[:n0_test], idx1[:n1_test]])
    train_idx = np.concatenate([idx0[n0_test:], idx1[n1_test:]])
    rng.shuffle(test_idx)
    rng.shuffle(train_idx)
    return train_idx, test_idx
```

Функция обучения логистической регрессии, для положительного класса применяется коэффициент pos\_weight для учета дисбаланса классов, реализована l2 регуляризация:

```
def train_logreg(X, y, l2_reg, lr, epochs, pos_weight):
    n, d = X.shape
    w = np.zeros(d)

    if pos_weight == "auto":
        n_pos = max(int(y.sum()), 1)
        n_neg = max(int((y == 0).sum()), 1)
        pw = n_neg / n_pos
    else:
        pw = float(pos_weight)

    sample_w = np.where(y == 1, pw, 1.0)

    for ep in range(epochs):
        p = sigmoid(X @ w)
        err = (p - y) * sample_w
        grad = (X.T @ err) / n

        if l2_reg > 0:
            grad[1:] += l2_reg * w[1:]

        w -= lr * grad

    return w, pw
```

Также реализована функция оценки результата по метрикам

## 4. Результат

TEST_SIZE=0.2, VAL_SIZE=0.2, L2_REG=0.05, EPOCHS=3000 POS_WEIGHT_MODE=auto									
seed= 0   test: n=335, pos=40   thr=0.65 pw=7.39   acc=0.872 rec1=0.500 prec1=0.465 f1_1=0.482   rec0=0.922 prec0=0.932 f1_0=0.927   TP=20 FP=23 FN=20 TN=272									
seed= 1   test: n=335, pos=40   thr=0.73 pw=7.39   acc=0.881 rec1=0.450 prec1=0.500 f1_1=0.474   rec0=0.939 prec0=0.926 f1_0=0.933   TP=18 FP=18 FN=22 TN=277									
seed= 2   test: n=335, pos=40   thr=0.65 pw=7.39   acc=0.839 rec1=0.575 prec1=0.383 f1_1=0.460   rec0=0.875 prec0=0.938 f1_0=0.905   TP=23 FP=37 FN=17 TN=258									
seed= 3   test: n=335, pos=40   thr=0.73 pw=7.39   acc=0.868 rec1=0.458 prec1=0.419 f1_1=0.434   rec0=0.915 prec0=0.925 f1_0=0.920   TP=18 FP=25 FN=22 TN=270									
seed= 4   test: n=335, pos=40   thr=0.69 pw=7.39   acc=0.875 rec1=0.500 prec1=0.476 f1_1=0.488   rec0=0.925 prec0=0.932 f1_0=0.929   TP=20 FP=22 FN=20 TN=273									
seed= 5   test: n=335, pos=40   thr=0.56 pw=7.39   acc=0.788 rec1=0.650 prec1=0.313 f1_1=0.423   rec0=0.807 prec0=0.944 f1_0=0.870   TP=26 FP=57 FN=14 TN=238									
seed= 10   test: n=335, pos=40   thr=0.65 pw=7.39   acc=0.848 rec1=0.525 prec1=0.396 f1_1=0.452   rec0=0.892 prec0=0.933 f1_0=0.912   TP=21 FP=32 FN=19 TN=263									
seed= 22   test: n=335, pos=40   thr=0.48 pw=7.39   acc=0.785 rec1=0.775 prec1=0.330 f1_1=0.463   rec0=0.786 prec0=0.963 f1_0=0.866   TP=31 FP=63 FN=9 TN=232									
seed= 42   test: n=335, pos=40   thr=0.69 pw=7.39   acc=0.881 rec1=0.525 prec1=0.500 f1_1=0.512   rec0=0.929 prec0=0.935 f1_0=0.932   TP=21 FP=21 FN=19 TN=274									
seed= 99   test: n=335, pos=40   thr=0.61 pw=7.39   acc=0.869 rec1=0.525 prec1=0.457 f1_1=0.488   rec0=0.915 prec0=0.934 f1_0=0.925   TP=21 FP=25 FN=19 TN=270									
СРЕДНЕЕ ПО СЕД (± std):									
Accuracy: 0.850 ± 0.034									
Recall(1): 0.548 ± 0.094									
Prec(1): 0.424 ± 0.064									
F1(1): 0.467 ± 0.026									
Recall(0): 0.891 ± 0.050									
Prec(0): 0.936 ± 0.010									
F1(0): 0.912 ± 0.023									

Видна относительно небольшое стандартное отклонение, что показывает что модель не является чувствительной к случайному разбиению данных

Accuracy 85% неплохой результат, но является второстепенным в виду дисбаланса классов/

Качество метрик для класса 1:

обнаруживает 55% реально ушедших сотрудников больше половины. но лишь 42% помеченных моделью как уходящие реально уходят, что тоже близко к половине

Для класса 0:

картина получше, модель уверенно распознаёт 90% сотрудников которые остаются. В 94% помеченных как остающиеся действительно остаются.

Модель уверенно распознает класс остающихся сотрудников, демонстрирует удовлетворительное качество по распознаванию класса "уход", с учетом дисбаланса классов.