Защита персональных данных клиентов

Описание проекта

Необходимо защитить данные клиентов страховой компании «Хоть потоп». Разработайте такой метод преобразования данных, чтобы по ним было сложно восстановить персональную информацию. Обоснуйте корректность его работы. Нужно защитить данные, чтобы при преобразовании качество моделей машинного обучения не ухудшилось. Подбирать наилучшую модель не требуется.

Признаки:

- пол
- возраст
- зарплата застрахованного
- количество членов его семьи.

Целевой признак

- количество страховых выплат клиенту за последние 5 лет.

B [1]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error as MSE
from sklearn.model_selection import train_test_split

import os
import urllib

pd.options.display.float_format = "{:,.3f}".format
```

Загрузка данныхlinear_model

B [2]:
```
datasets = {
    'name':'datasets/insurance.csv',\
        'url': 'https://code.s3.yandex.net/datasets/insurance.csv'
}

os.makedirs(datasets['name'].split('/')[0], exist_ok=True)
_ = urllib.request.urlretrieve(datasets['url'], datasets['name'])
```

B [3]:
```
data = pd.read_csv(datasets['name'])
```

In [4]:
```
data.sample(3)
```

Out[4]:

	Пол	Возраст	Зарплата	Члены семьи	Страховые выплаты
4234	0	29.000	38,900.000	1	0
4128	1	25.000	34,000.000	1	0
4368	0	56.000	50,300.000	2	3

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Пол                5000 non-null   int64
 1   Возраст            5000 non-null   float64
 2   Зарплата           5000 non-null   float64
 3   Члены семьи        5000 non-null   int64
 4   Страховые выплаты  5000 non-null   int64
dtypes: float64(2), int64(3)
memory usage: 195.4 KB
```

In [6]:
```
data.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	m
Пол	5,000.000	0.499	0.500	0.000	0.000	0.000	1.000	1.(
Возраст	5,000.000	30.953	8.441	18.000	24.000	30.000	37.000	65.(
Зарплата	5,000.000	39,916.360	9,900.084	5,300.000	33,300.000	40,200.000	46,600.000	79,000.(
Члены семьи	5,000.000	1.194	1.091	0.000	0.000	1.000	2.000	6.(
Страховые выплаты	5,000.000	0.148	0.463	0.000	0.000	0.000	0.000	5.(

In [7]:
```
features = data.drop(['Страховые выплаты'], axis=1)
target = data['Страховые выплаты']
```

Умножение матриц

Признаки умножают на обратимую матрицу. Изменится ли качество линейной регрессии?

Обоснование

Задача предсказания (лог регр) формулируется матрично в след виде

$$a = Xw$$

Обучение модели: минимизация функции потерь (метрики `MSE`)

$$w = \arg\min_w MSE(Xw, y)$$

Матрица весов находится (в процессе "обучения") по след выражению

$$w = [X^T X]^{-1} X^T y$$

Хотим посмотреть, повлияет ли линейное преобразование признаков (с последующем обучением модели) на метрику качества на предсказании. Подставим в предыдущее выражение вместо матрицы признаков результат линейного преобразования (произведение матрицы признаков на обратимую квадратную матрицу). Если матрица признаков размерности `m, n` домножив её на обратимую матрицу размерности `n, n`, то в результате получим матрицу `m, n`.

$$w' = [(XM)^T (XM)]^{-1} (XM)^T y$$

Воспользуемся тем, что:

$$[AB]^T = B^T A^T$$

$$w' = [M^T X^T XM]^{-1} M^T X^T y$$

Так как в выражении

$$M^T X^T XM = M^T (X^T XM)$$

первый множитель имеет размерность `n, n`, как и второй `n, n`, то

$$w' = [X^T XM]^{-1} [M^T]^{-1} M^T X^T y$$

Так как

$$[AB]^{-1} = B^{-1} A^{-1}$$

$$w' = [X^T XM]^{-1} [MM^{-1}]^T X^T y = [X^T XM]^{-1} E^T X^T y = [X^T XM]^{-1} X^T y$$

$$w' = M^{-1} [X^T X]^{-1} X^T y$$

Возвращаясь к самой первой формуле, получаем

$$\begin{cases} a = Xw = X[X^T X]^{-1} X^T y \\ a' = XMw = XMM^{-1} [X^T X]^{-1} X^T y \end{cases}$$

$$\begin{cases} a = X[X^T X]^{-1} X^T y \\ a' = XE[X^T X]^{-1} X^T y \end{cases}$$

Т е от линейного преобразования матрицы признаков предсказания не поменяются (в связи с подходом к обучению модели). Как следствие качество модели `не поменяется`.

$$\begin{cases} w = [X^T X]^{-1} X^T y \\ w' = M^{-1} [X^T X]^{-1} X^T y \end{cases}$$

$$w' = M^{-1} w$$

B [8]:
```
model = LinearRegression()
```

B [9]:
```
model.fit(features, target)
preds_base = model.predict(features)
r2_base = r2_score(target, preds_base)
mse_base = MSE(target, preds_base)
```

B [10]:
```
mtx = np.random.rand(features.shape[1], features.shape[1])
mtx.shape
```

Out[10]:
```
(4, 4)
```

B [11]:
```
try:
    np.linalg.inv(mtx)
except:
    print('Матрица mtx необратимая')
```

B [12]:
```
new_features = features @ mtx
```

B [13]:
```
model.fit(new_features, target)
preds_new = model.predict(new_features)
r2_new = r2_score(target, preds_new)
mse_new = MSE(target, preds_new)
```

B [14]:
```
print('Метрика качества такая же, ч т д'\
      if round(r2_base - r2_new, 3) < 1E-10 else\
      'Метрика различается по сравнению с первоначальной')
```

```
Метрика качества такая же, ч т д
```

Алгоритм преобразования и его применение на практике

Обоснование и описание метода преобразования исходных признаков

По сути весь алгоритм был описан выше - линейное преобразование признаков не приводит к изменению предсказаний линейной регрессии, что в свою очередь даёт нам возвожность воспользоваться матрицей случайных чисел размрностью `features.shape[1]` для получения новой матрицы признаков ("закодированой"). Конечно, все вышеперечисленное имеет место, если матрица обратимые

B [15]:
```
mtx = np.random.rand(features.shape[1], features.shape[1])
try:
    np.linalg.inv(mtx)
except:
    print('Матрица mtx необратимая')
```

B [16]:
```
features_train, features_test, target_train, target_test=\
    train_test_split(features, target, random_state=1, test_size=.25)
```

B [17]:
```
model = LinearRegression()
model.fit(features_train, target_train)
preds_base = model.predict(features_test)
r2_base = r2_score(target_test, preds_base)
```

B [18]:
```
model = LinearRegression()
model.fit(features_train, target_train)
preds_upd = model.predict(features_test)
r2_upd = r2_score(target_test, preds_upd)
```

B [24]:
```
if round(r2_base - r2_upd, 3) < 1E-10 and\
          round(mse_base - mse_new, 3) < 1E-10:
    print('''Метрики MSE (в данной задаче функция потерь) и R2_score\n'''
          '''не изименились от кодирования признаков''')
else:
    print('Метрики различаются по сравнению с первоначальными оценками')
```

```
Метрики MSE (в данной задаче функция потерь) и R2_score
не изименилась от кодирования признаков
```

Итоговый вывод

Проект позволил лучше понять суть алгоритма линейной регрессии. На практике убедились в том, что веса действительно определяются лишь обычными матричными операциями, что на мой взгляд, удивительно (без каких либо алгоритмов обучения, просто математическое выражение) - удалось достичь понимания данного выражения с помощью кодирования (линейного преобразования исходной матрицы признаков). Получив то, что от кодирования метрики не меняются, можем сделать вывод в правильности (если точнее, не можем отрицать их действенности) исходных формул и математических преобразований.