Content

Project Description

Insurance company would like to develop a method of protection of client's personal data. It's required to have an option to get the original information (unprotected personal data) after the completion of such protection. The quality of the model prediction shall be the same on the original and protected data. The protection of the personal data shall be done using the matrix operations.

Main tasks of the project are:

- to import and overview data, select features and target;
- to develop algorithm of data protection;
- validate the algorithm and apply it on the provided data;

Data import

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
```

In [2]:
```
dataset = pd.read_csv('insurance.csv')
```

In [3]:
```
# display first 5 rows
dataset.head()
```

Out[3]:

	Пол	Возраст	Зарплата	Члены семьи	Страховые выплаты
0	1	41.0	49600.0	1	0
1	0	46.0	38000.0	1	1
2	0	29.0	21000.0	0	0
3	0	21.0	41700.0	2	0
4	1	28.0	26100.0	0	0

In [4]:
```
# display the data distribution
dataset.describe()
```

Out[4]:

	Пол	Возраст	Зарплата	Члены семьи	Страховые выплаты
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	0.499000	30.952800	39916.360000	1.194200	0.148000
std	0.500049	8.440807	9900.083569	1.091387	0.463183
min	0.000000	18.000000	5300.000000	0.000000	0.000000
25%	0.000000	24.000000	33300.000000	0.000000	0.000000
50%	0.000000	30.000000	40200.000000	1.000000	0.000000
75%	1.000000	37.000000	46600.000000	2.000000	0.000000
max	1.000000	65.000000	79000.000000	6.000000	5.000000

In [5]:
```
# display dataset info
dataset.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]:
```
# Selection of target
target = dataset['Страховые выплаты']
```

In [7]:
```
# selection of features
features = dataset.drop(columns = 'Страховые выплаты')
```

Conclusion:

- Dataset imported and has 5000 rows and 5 columns: sex, age, salary, quantity of persons in family and insurance payment.
- Dataset splat on target and features. Target is column 'Страховые выплаты'.
- After import, overview and selection of target and features it's possible to move on the next step.

Multiplication of matrix

Notations:

- X — Features matrix (zero column has only zeros)
- y — target vector
- P — matrix, used for multiplication
- w — vector of weights of linear regression (zer0 element is equal to shift)

Proof:

$$a = Xw = XEw = XPP^{-1}w = (XP)P^{-1}w = (XP)w'$$

$$w = (X^T X)^{-1} X^T y$$

$$w' = ((XP)^T XP)^{-1} (XP)^T y$$

$$w' = (P^T (X^T X) P)^{-1} (XP)^T y$$

$$w' = (P^T (X^T X) P)^{-1} P^T X^T y$$

$$w' = P^{-1} (X^T X)^{-1} (P^T)^{-1} P^T X^T y$$

$$w' = P^{-1} (X^T X)^{-1} E X^T y$$

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

Question

Features to be multiplied on inverse matrix. Will affect on the linear regression quality?

a. Quality to be changed. Provide the example of such matrix.

b. Quality will remain the same. Specify how does the features of linear regression in multiplied matrix and original are related.

Answer on the question on example below:

In [8]:
```
# matrix generation
matrix_generated = np.random.randint(10, size=(4, 4))

print(matrix_generated)
```

```
[[8 6 9 1]
 [4 1 9 2]
 [3 0 0 0]
 [3 6 9 9]]
```

In [9]:
```
# check does the generation matrix is invertable
matrix_generated_inv = np.linalg.inv(matrix_generated)
matrix_generated_inv
```

Out[9]:
```
array([[ 2.77555756e-17,  2.77555756e-17,  3.33333333e-01,
         3.46944695e-18],
       [ 1.75000000e-01, -2.00000000e-01, -2.25000000e-01,
         2.50000000e-02],
       [ 8.33333333e-03,  1.33333333e-01, -1.69444444e-01,
        -3.05555556e-02],
       [-1.25000000e-01,  0.00000000e+00,  2.08333333e-01,
         1.25000000e-01]])
```

In [10]:
```
# Selection 5 rows of datset
example_df = features.loc[:5,:]
example_df
```

Out[10]:

	Пол	Возраст	Зарплата	Члены семьи
0	1	41.0	49600.0	1
1	0	46.0	38000.0	1
2	0	29.0	21000.0	0
3	0	21.0	41700.0	2
4	1	28.0	26100.0	0
5	1	43.0	41000.0	2

In [11]:
```
# selection of 5 targets for these features
example_target = target[:6]
example_target
```

Out[11]:
```
0    0
1    1
2    0
3    0
4    0
5    1
Name: Страховые выплаты, dtype: int64
```

In [12]:
```
# linear regression trainig and disply the score
model = LinearRegression()
model.fit(example_df, example_target)
print(model.score(example_df, example_target))
```

```
0.9406607877645261
```

Score of Linear regression on the sample data is 0.94

In [13]:
```
# multiply the features on generated matrix
example_df_multiplied = example_df @ matrix_generated
```

In [14]:
```
example_df_multiplied
```

Out[14]:

	0	1	2	3
0	148975.0	53.0	387.0	92.0
1	114187.0	52.0	423.0	101.0
2	63116.0	29.0	261.0	58.0
3	125190.0	33.0	207.0	60.0
4	78420.0	34.0	261.0	57.0
5	123186.0	61.0	414.0	105.0

In [15]:
```
# training of model on trasnformed data
model_inv = LinearRegression()
model_inv.fit(example_df_multiplied, example_target)
print(model_inv.score(example_df_multiplied, example_target))
```

```
0.940660787764526
```

Score of Linear regression on the transformed data is 0.94

In [16]:
```
# check to possibility to get original data after transformation
round(example_df_multiplied@matrix_generated_inv,0)
```

Out[16]:

	0	1	2	3
0	1.0	41.0	49600.0	1.0
1	0.0	46.0	38000.0	1.0
2	0.0	29.0	21000.0	0.0
3	0.0	21.0	41700.0	2.0
4	1.0	28.0	26100.0	0.0
5	1.0	43.0	41000.0	2.0

Answer: Quality will remain the same.

Validation: Features of original and transformed matrixes are different only in the multiplication on the generated matrix.

example of matrixes multiplication:

```
a b
c d

и

1 2
3 4

Transformed matrix:

a1+b3 a2+b4
c1+d3 c2+d4
```

Transformation algorithm

Algorithm

1) Generation of random matrix with length equal to features width.

2) Test of generated matrix to be invertible. If matrix invertible - matrix is accepted, if not, the new matrix to be generated until the we will not get the invertible matrix.

3) Features to be multiplied on generated matrix.

Validation

After the multiplication of features on invertible matrix there is always an option to get the original data, multiplied the transformed feature on inverted generated matrix.

The data in dataset to be protected, due to the fact that without generated matrix there will be no way get the original data. The reversed transformation is possible due to the fact that if generated matrix will be multiplied on inverted matrix we will get the unity matrix - E.

Algorithm testing

In [17]:
```
temp = 0
while temp == 0:
    try:
        matrix_generated = np.random.randint(10, size=(4, 4))
        matrix_generated_inv = np.linalg.inv(matrix_generated)
        temp = matrix_generated_inv[0,0]
    except:
        pass
```

Prediction on transformed features

In [18]:
```
model_inv  = LinearRegression()
model_inv.fit(features@matrix_generated, target)
print(model_inv.score(features@matrix_generated, target))
```

```
0.42494550286668176
```

Prediction on original data

In [19]:
```
model = LinearRegression()
model.fit(features, target)
print(model.score(features, target))
```

```
0.42494550286668
```

In [20]:
```
if round(model_inv.score(features@matrix_generated, target),3) == round(model.score(features, target),3):
    print('Models scores are equal!')
else:
    print('Models scores are different!')
```

```
Models scores are equal!
```

Conclusion

- Data successfully imported, features and target selected. Target columns is 'Страховые выплаты'.
- The data protection algorithm developed - multiplication of features on invertible matrix allow to protect the data.
- The algorithm has positive result during testing - the quality of model are remain the same on the original ans transformed data.