Final_Project

April 2, 2024

Project Description

Prudential, one of the largest issuers of life insurance in the USA, is hiring passionate data scientists to join a newly-formed Data Science group dedicated to solving complex challenges and identifying opportunities. The results achieved so far have been impressive, but we aspire for more.

In a one-click shopping world with on-demand everything, the life insurance application process remains antiquated. Customers provide extensive information to determine risk classification and eligibility, which includes scheduling medical exams—a process that typically spans an average of 30 days.

The consequence? People are deterred. This is evidenced by the fact that only 40% of U.S. households own individual life insurance policies. Prudential aims to expedite and streamline the application process for both new and existing customers, while upholding privacy boundaries.

By developing a predictive model that accurately classifies risk through a more automated approach, you can significantly influence the public perception of the industry.

The results obtained will enable Prudential to gain a deeper understanding of the predictive power of the existing assessment's data points, thus facilitating a substantial streamlining of the process.

In this project, you are provided with over a hundred variables describing attributes of life insurance applicants. The objective is to predict the "**Response**" variable for each `ID` in the test set. `Response` is an ordinal measure of risk, which consists of 8 levels.

File descriptions

- `train.csv` - the training set you may use to train a predictive model.
- `test.csv` - the test set to evaluate the model constructed using the training data. The test set must not be used at any stage of the training.

Variable	Description
Id	A unique identifier associated with an application
Product_Info_1-7	A set of normalized variables relating to the product applied for
Ins_Age	Normalized age of applicant
Ht	Normalized height of applicant
Wt	Normalized weight of applicant
BMI	Normalized BMI of applicant
Employment_Info_1-6	A set of normalized variables relating to the employment history of the applicant

Variable	Description
InsuredInfo_1-6	A set of normalized variables providing information about the applican
Insurance_History_1-9	A set of normalized variables relating to the insurance history of the applicant
Family_Hist_1-5	A set of normalized variables relating to the family history of the applicant
Medical_History_1-41	A set of normalized variables relating to the medical history of the applicant
Medical_Keyword_1-48	A set of dummy variables relating to the presence of/absence of a medical keyword being associated with the application
Response	Target variable, an ordinal variable relating to the final decision associated with an application

The following variables are all categorical (nominal):

Product_Info_1, Product_Info_2, Product_Info_3, Product_Info_5, Product_Info_6, Product_Info_7, Employment_Info_2, Employment_Info_3, Employment_Info_5, InsuredInfo_1, InsuredInfo_2, InsuredInfo_3, InsuredInfo_4, InsuredInfo_5, InsuredInfo_6, InsuredInfo_7, Insurance_History_1, Insurance_History_2, Insurance_History_3, Insurance_History_4, Insurance_History_7, Insurance_History_8, Insurance_History_9, Family_Hist_1, Medical_History_2, Medical_History_3, Medical_History_4, Medical_History_5, Medical_History_6, Medical_History_7, Medical_History_8, Medical_History_9, Medical_History_11, Medical_History_12, Medical_History_13, Medical_History_14, Medical_History_16, Medical_History_17, Medical_History_18, Medical_History_19, Medical_History_20, Medical_History_21, Medical_History_22, Medical_History_23, Medical_History_25, Medical_History_26, Medical_History_27, Medical_History_28, Medical_History_29, Medical_History_30, Medical_History_31, Medical_History_33, Medical_History_34, Medical_History_35, Medical_History_36, Medical_History_37, Medical_History_38, Medical_History_39, Medical_History_40, Medical_History_41

The following variables are continuous:

Product_Info_4, Ins_Age, Ht, Wt, BMI, Employment_Info_1, Employment_Info_4, Employment_Info_6, Insurance_History_5, Family_Hist_2, Family_Hist_3, Family_Hist_4, Family_Hist_5

The following variables are discrete:

Medical_History_1, Medical_History_10, Medical_History_15, Medical_History_24, Medical_History_32

Dummy variables:

Medical_Keyword_1-48 are dummy variables.

Evaluation

Submissions are scored based on the *quadratic weighted kappa*, which measures the agreement between two ratings. This metric typically varies from 0 (random agreement) to 1 (complete agreement). In the event that there is less agreement between the raters than expected by chance, this metric may go below 0.

The response variable has **8** possible ratings. Each application is characterized by a tuple (e_a, e_b), which corresponds to its scores by Rater A (*actual risk*) and Rater B (*predicted risk*). The quadratic weighted kappa is calculated as follows.

First, an N-by-N histogram matrix O is constructed, such that $O_{i,j}$ corresponds to the number of applications that received a rating i by A and a rating j by B.

Next, an N-by-N matrix of weights, w, is calculated based on the difference between raters' scores:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2}.$$

An N-by-N histogram matrix of expected ratings, E, is calculated, assuming that there is no correlation between rating scores. This is calculated as the outer product between each rater's histogram vector of ratings, normalized such that E and O have the same sum.

From these three matrices, the quadratic weighted kappa is calculated as:

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}.$$

Apply the classification methods, including model selection techniques for choosing hyperparameters, learned from this course (or elsewhere), to train a predictive model using the training set.

Use your model to predict the risk scores (`Response`) in the test set.

Evaluate the performance of your model using the aforementioned evaluation method (**quadratic weighted kappa**).

[1]:
```python
import numpy as np
import pandas as pd

train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
print('Train data shape: ', train_df.shape)
print('Test data shape: ', test_df.shape)
print('Risk scores:', train_df['Response'].unique())
print('--' *40)
print('Columns:', train_df.columns)
print('--' *40)
```

```
Train data shape:  (47504, 128)
Test data shape:  (11877, 128)
Risk scores: [7 8 2 4 6 1 3 5]
--------------------------------------------------------------------------------
Columns: Index(['Id', 'Product_Info_1', 'Product_Info_2', 'Product_Info_3',
       'Product_Info_4', 'Product_Info_5', 'Product_Info_6', 'Product_Info_7',
       'Ins_Age', 'Ht',
       …
       'Medical_Keyword_40', 'Medical_Keyword_41', 'Medical_Keyword_42',
       'Medical_Keyword_43', 'Medical_Keyword_44', 'Medical_Keyword_45',
       'Medical_Keyword_46', 'Medical_Keyword_47', 'Medical_Keyword_48',
       'Response'],
      dtype='object', length=128)
--------------------------------------------------------------------------------
```

[2]:
```
# randomly sample 20 records from the training data
# A is the vector of actual risk scores for the 20 records
A = train_df.sample(20).Response.values

# B is a vector of 20 randomly predicted risk scores
B = np.random.randint(1, 9, size=20)
pd.DataFrame(np.c_[(A, B)], columns=['A', 'B'])
```

[2]:
```
    A  B
0   5  3
1   6  6
2   6  5
3   6  8
4   1  4
5   2  6
6   1  7
7   6  8
8   8  1
9   8  1
10  1  2
11  1  6
12  6  7
13  1  2
14  8  7
15  8  6
16  8  8
17  5  6
18  1  6
19  8  2
```

[3]:
```
hist_matrix, _, _ = np.histogram2d(A, B, bins=8)
O = pd.DataFrame(hist_matrix, index=range(1, 9), columns=range(1, 9),␣
 ↪dtype='int')
O
```

[3]:
```
   1  2  3  4  5  6  7  8
1  0  2  0  1  0  2  1  0
2  0  0  0  0  0  1  0  0
3  0  0  0  0  0  0  0  0
4  0  0  0  0  0  0  0  0
```

```
5  0  0  1  0  0  1  0  0
6  0  0  0  0  1  1  1  2
7  0  0  0  0  0  0  0  0
8  2  1  0  0  0  1  1  1
```

[4]:
```
w = np.zeros((8, 8))
for i in range(8):
    for j in range(8):
        w[i, j] = (((i+1) - (j+1)) ** 2) / ((8 - 1) ** 2)
w = pd.DataFrame(w, index=range(1, 9), columns=range(1, 9))
w
```

[4]:
```
          1         2         3         4         5         6         7  \
1  0.000000  0.020408  0.081633  0.183673  0.326531  0.510204  0.734694
2  0.020408  0.000000  0.020408  0.081633  0.183673  0.326531  0.510204
3  0.081633  0.020408  0.000000  0.020408  0.081633  0.183673  0.326531
4  0.183673  0.081633  0.020408  0.000000  0.020408  0.081633  0.183673
5  0.326531  0.183673  0.081633  0.020408  0.000000  0.020408  0.081633
6  0.510204  0.326531  0.183673  0.081633  0.020408  0.000000  0.020408
7  0.734694  0.510204  0.326531  0.183673  0.081633  0.020408  0.000000
8  1.000000  0.734694  0.510204  0.326531  0.183673  0.081633  0.020408

          8
1  1.000000
2  0.734694
3  0.510204
4  0.326531
5  0.183673
6  0.081633
7  0.020408
8  0.000000
```

[5]:
```
hist_A, _ = np.histogram(A, bins=range(1, 10))
hist_B, _ = np.histogram(B, bins=range(1, 10))
E = pd.DataFrame(np.outer(hist_A, hist_B),
                 index=range(1, 9), columns=range(1, 9), dtype='int')
E = E * O.sum().sum() / E.sum().sum()
E
```

[5]:
```
     1     2     3     4     5    6     7     8
1  0.6  0.90  0.30  0.30  0.30  1.8  0.90  0.90
2  0.1  0.15  0.05  0.05  0.05  0.3  0.15  0.15
3  0.0  0.00  0.00  0.00  0.00  0.0  0.00  0.00
4  0.0  0.00  0.00  0.00  0.00  0.0  0.00  0.00
5  0.2  0.30  0.10  0.10  0.10  0.6  0.30  0.30
6  0.5  0.75  0.25  0.25  0.25  1.5  0.75  0.75
7  0.0  0.00  0.00  0.00  0.00  0.0  0.00  0.00
```

```
8  0.6  0.90  0.30  0.30  0.30  1.8  0.90  0.90
```

[6]:
```
kappa = 1 - (w*O).sum().sum() / (w*E).sum().sum()
kappa
```

[6]:
```
0.023408924652523755
```

[7]:
```
# define a function to calculate the quadratic weighted kappa score
def quadratic_weighted_kappa(A, B, N=8):
    '''
    A, B: numpy arrays with integer values between 1 and N
    '''
    O, _, _ = np.histogram2d(A, B, bins=N)
    w = np.zeros((N, N))
    for i in range(N):
        for j in range(N):
            w[i, j] = ((i - j) ** 2) / ((N - 1) ** 2)
    hist_A, _ = np.histogram(A, bins=range(1, N+2))
    hist_B, _ = np.histogram(B, bins=range(1, N+2))
    E = np.outer(hist_A, hist_B)
    E = E * O.sum() / E.sum()
    return 1 - (w*O).sum() / (w*E).sum()
```

[8]:
```
quadratic_weighted_kappa(A, B)
```

[8]:
```
0.023408924652523755
```

[9]:
```
quadratic_weighted_kappa(A, A)
```

[9]:
```
1.0
```

[ ]:
```

```