UNIVERSITY OF ECONOMICS AND LAW

Nguyễn Tuấn Hưng	UEL
Lâm Nhựt Thịnh	UEL
Hoàng Phước Thành	FPT
Nguyễn Đức Minh Tấn	UEL
Thái Tuấn Kha	UEL

---

1. Data cleaning and EDA

- The initial step is loading the train data and test data. Here the train data has 48030 rows and 65 columns, while the test one has 5000 rows and 64 columns (minus column 'label').

```
train_data.head()
✓ 0.1s                                                        Python
```

id	label	time_1	time_2	Field_11	cat_1	cat_2	cat_3	cat_4	cat_5	date_1	...	unknown_var_19	unknown_var_20	social_friend_count
11651	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	285.0
48491	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.0	2.0	2534.0
42868	0	2017-01-23T18:00:32.69Z	2017-01-23T18:00:32.69Z	NaN	C2	P1	1.0	NaN	1.0	12/27/2012	...	NaN	NaN	653.0
68835	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	0.0
73688	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	0.0

5 rows × 65 columns

```
test_data.head()
✓ 0.4s                                                        Python
```

id	time_1	time_2	Field_11	cat_1	cat_2	cat_3	cat_4	cat_5	date_1	mer_des	...	unknown_var_19	unknown_var_20	social_friend_count
31502	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	NaT	NaN	...	NaN	NaN	0.0
52725	2019-03-15T07:27:31.398Z	2019-03-19T02:42:37Z	NaT	C1	P2	1.0	NaN	1.0	2019-03-19	NaN	...	0.0	2.0	NaN
31752	2017-02-17T11:51:53.373Z	2017-02-17T11:51:53.373Z	2016-11-01	C1	P2	1.0	UI	1.0	2016-11-01	NaN	...	NaN	NaN	NaN
78992	2019-02-28T01:27:45.783Z	2019-02-28T03:06:39Z	NaT	C1	P2	1.0	NaN	1.0	2019-02-28	NaN	...	2.0	6.0	NaN
52965	2019-08-12T10:16:03.498Z	2019-08-15T01:48:17Z	NaT	C1	P2	1.0	NaN	1.0	2019-08-15	NaN	...	1.0	1.0	0.0

5 rows × 64 columns

- The columns' names seem to be poorly formatted, and there are extra special characters in the data so we will be formatting these

```
for (columnName, columnData) in train_data.iteritems():
    train_data[columnName] = columnData.replace(['-    ', ' -    ', ' '], '')

for (columnName, columnData) in test_data.iteritems():
    test_data[columnName] = columnData.replace(['-    ', ' -    ', ' '], '')

train_data.columns = train_data.columns.str.replace(' ', '')
test_data.columns = test_data.columns.str.replace(' ', '')
✓ 0.4s
```

- Afterwards, we drop any columns in the train_data with more than 60% null values. We drop the same columns in the test_data.

```
columns_to_drop = [col for col in train_data.columns if (train_data[col].isna().sum()/len(train_data[col]) > 0.6)]
train_data = train_data.drop(columns=columns_to_drop, axis=1)
test_data = test_data.drop(columns=columns_to_drop, axis=1)
print(train_data.info())
print(test_data.info())
✓ 0.2s                                                        Python
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48030 entries, 11651 to 89883
Data columns (total 53 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   label                  48030 non-null  int64
 1   time_1                 22991 non-null  object
 2   time_2                 22991 non-null  object
 3   cat_1                  22991 non-null  object
 4   cat_2                  22991 non-null  object
 5   cat_3                  22991 non-null  float64
 6   cat_5                  22991 non-null  float64
 7   mul_rate               22991 non-null  float64
 8   value                  22991 non-null  object
 9   cat_6                  22991 non-null  float64
…
 44  unknown_var_15         22991 non-null  float64
 45  unknown_var_16         26433 non-null  float64
 46  unknown_var_18         21405 non-null  float64
 47  unknown_var_19         21405 non-null  float64
 48  unknown_var_20         21405 non-null  float64
 49  social_friend_count    21644 non-null  float64
 50  social_sex_info        21058 non-null  object
 51  social_subcriber_count 21644 non-null  float64
 52  social_location_id     21644 non-null  object
dtypes: float64(29), int64(2), object(22)
```

- Next, we will drop any categorical features with more than 4 unique values. Features with exception will be formatted for further assessment.

```python
# get list of categorical features
list_categorical_cols = list(train_data.columns[train_data.dtypes == 'O'])

categorical_cols_exceptions = ['value', 'review_value']

print(list_categorical_cols)
# get info about different categories
for cat_feature in list_categorical_cols:
    if train_data[cat_feature].value_counts().shape[0] > 4 and cat_feature not in categorical_cols_exceptions:
        train_data = train_data.drop(cat_feature, axis=1)
    else:
        print(train_data[cat_feature].value_counts())

list_categorical_cols_test = list(test_data.columns[test_data.dtypes == 'O'])
for cat_feature in list_categorical_cols_test:
    if test_data[cat_feature].value_counts().shape[0] > 4 and cat_feature not in categorical_cols_exceptions:
        test_data = test_data.drop(cat_feature, axis=1)
    else:
        print(test_data[cat_feature].value_counts())
```
✓ 0.4s Python

```
['time_1', 'time_2', 'cat_1', 'cat_2', 'value', 'review_value', 'date_2', 'date_3', 'sex', 'address', 'location_id', 'mer_id', 'mer_name', 'trans_location', 'cat_8', 'job', 'cat_9', 'com_type', 'cat_12', 'unknown_var_5', 'social_sex_info', 'social_location_id']
C1    12406
C2    10585
Name: cat_1, dtype: int64
P2    12512
P1    10479
Name: cat_2, dtype: int64
1,490,000    1509
1,390,000    1235
          ...
```

```
3,036,000      1
4,467,200      1
Name: value, Length: 4404, dtype: int64
             21019
1,150,000      205
             ...
3,177,280        1
2,221,000        1
Name: review_value, Length: 583, dtype: int64
MALE      13149
FEMALE     9842
Name: sex, dtype: int64
male      12465
female     8593
Name: social_sex_info, dtype: int64
C1    1365
C2    1208
Name: cat_1, dtype: int64
P2    1366
P1    1207
Name: cat_2, dtype: int64
MALE      1536
FEMALE    1037
Name: sex, dtype: int64
male      1310
female     883
Name: social_sex_info, dtype: int64
```

- We find that test_data did not drop date_2 and date_3 while train_data did. So for the sake of conformity, we will drop those in test_data.

```python
test_data = test_data.drop(columns=["date_2", "date_3"], axis=1)
```
✓ 0.3s Python

- We now format data in value and review_value in train_data, and test_data is already formatted so we will ignore it.

```python
train_data['value'] = pd.to_numeric(train_data['value'].str.replace(',', ''))
train_data['review_value'] = pd.to_numeric(train_data['review_value'].str.replace(',', ''))
```
✓ 0.7s Python

- We then drop duplicates from both train_data and test_data.

```python
new_train = train_data.groupby(train_data.index).first()

cols = list(train_data.columns)
train_data.loc[train_data.index.isin(new_train.index), cols] = new_train[cols]

new_test = test_data.groupby(test_data.index).first()

cols = list(test_data.columns)
test_data.loc[test_data.index.isin(new_test.index), cols] = new_test[cols]
```
✓ 0.2s Python

- Since there are many missing values in the dataset, we will apply imputer using SimpleImputer to fill in missing values.

```python
def imputer(df):
    mean = SimpleImputer(missing_values=np.nan,strategy='mean')
    mode = SimpleImputer(missing_values=np.nan,strategy='most_frequent')
    for col in df.columns:
        if df[col].isnull().any()==True:
            if df[col].dtypes=='O':
                df[col] = df[col].apply(lambda x: col + '_No' if x is None else x)
            else:
                df[col]=mode.fit_transform(df[[col]])
    return df
```
✓ 0.4s Python

```
       label    cat_1     cat_2  cat_3  cat_5  mul_rate      value  cat_6  \
id
11651      0       C1        P2    1.0    1.0      0.00  4357542.0    1.0
48491      0       C2        P1    1.0    1.0      0.00  4480000.0    1.0
...      ...      ...       ...    ...    ...       ...        ...    ...
89826      1  cat_1_No  cat_2_No    1.0    1.0      0.00  1490000.0    1.0
89883      0       C2        P1    1.0    1.0      0.00  5000000.0    1.0

       num_date_review  review_value  ...  unknown_var_13 unknown_var_14  \
id                                    ...
11651              0.0     1150000.0  ...            0.52          0.500
48491              3.0     1150000.0  ...            0.60          0.570
...                ...           ...  ...             ...            ...
89826              0.0     1150000.0  ...            0.08          0.060
89883             20.0     1150000.0  ...            0.77          0.470

       unknown_var_15  unknown_var_16  unknown_var_18  unknown_var_19  \
id
11651            0.04             1.0             2.0             1.0
48491            0.06             1.0             2.0             0.0
...               ...             ...             ...             ...
89826            0.12             1.0             4.0             0.0
89883            0.60             1.0             4.0             2.0

       unknown_var_20  social_friend_count  social_sex_info  \
id
11651             2.0                285.0             male
48491             2.0               2534.0           female
42868             4.0                653.0           female
...               ...                  ...              ...
89649             4.0                 24.0             male
89826             4.0                 65.0           female
89883             4.0                  0.0           female

       social_subcriber_count
id
11651                    64.0
48491                   391.0
73688                     0.0
...                       ...
89649                     0.0
89826                     0.0
89883                   614.0

[48030 rows x 37 columns]
```

- Now we encode the remaining categorical features.

```python
train_data = pd.get_dummies(train_data)
test_data = pd.get_dummies(test_data)
```

✓ 0.6s Python

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 48030 entries, 11651 to 89883
Data columns (total 45 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   label                               48030 non-null  int64
 1   cat_3                               48030 non-null  float64
 2   cat_5                               48030 non-null  float64
 3   mul_rate                            48030 non-null  float64
 4   value                               48030 non-null  float64
 5   cat_6                               48030 non-null  float64
 …
 39  sex_FEMALE                          48030 non-null  uint8
 40  sex_MALE                            48030 non-null  uint8
 41  sex_sex_No                          48030 non-null  uint8
 42  social_sex_info_female              48030 non-null  uint8
 43  social_sex_info_male                48030 non-null  uint8
 44  social_sex_info_social_sex_info_No  48030 non-null  uint8
dtypes: float64(31), int64(2), uint8(12)
```

- With the continuous features, we use hist plots and scatter plots to see the distribution of these features. The raw distribution of these features are very skewed, so we try our best to transform some features to better visualize it.

```python
for feature in continuous_features:
    plt.figure(figsize=(20,8))
    plt.subplot(121)
    sns.histplot(x=feature,data=train_data)
    plt.show()
```

Python

- We illustrate the outliers in continuous features. Clearly, there are outliers in most features.

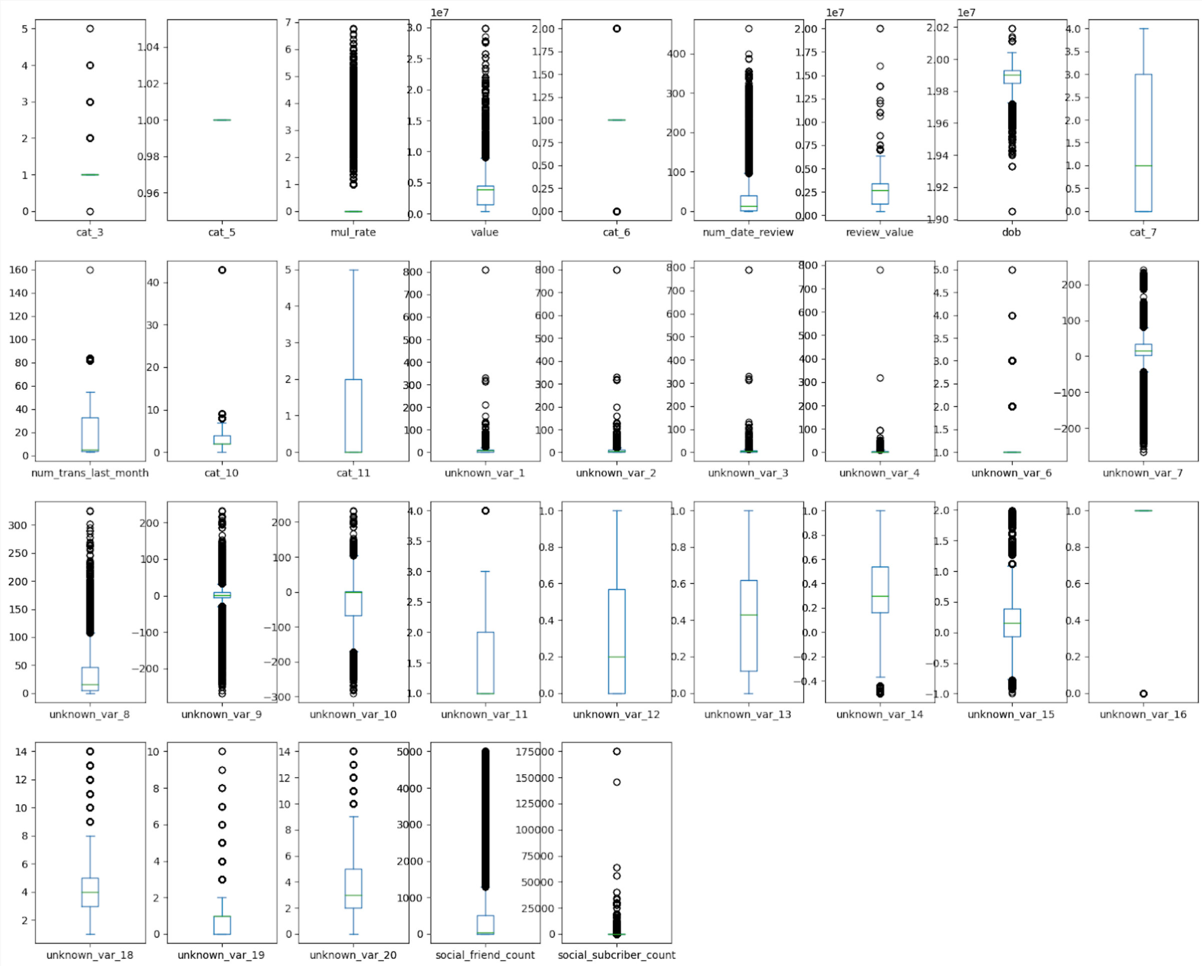


Figure 1: Box plot of features

2. Model deployment:

2.1 Analysis of problem requirements:

The problem requires the deployment of machine learning models to identify fraudulent transactions based on the dataset provided. Based on that requirement, the research team decided to implement the model in the following way:

- The recall of label 1 (this label is assigned to fraudulent transactions) is the most interested and preferred indicator. The higher the recall, the higher the percentage of fraudulent transactions detected, which coincides with the requirements of the problem.
- The next priority indicator is the precision of label 1. In fact, The recall of label 1 reaching high or even maximum is not difficult to achieve because it can be achieved by predicting most of the transactions in the data set are fraudulent. However, this leads to another dilemma that businesses will spend a lot of time, cost and human resources to monitor, evaluate and prevent transactions that the model considers to be fraudulent but in fact is not fraudulent, in addition, businesses also face the risk of losing customers with good credit. When the model predicts that all transactions in the data set are considered to be fraudulent (label 1), the precision of label 1 is now the percentage of the number of fraudulent transactions on the entire number of transactions (about 34.02%). The model needs to produce a label 1 precision as high as 34.02% as possible.

- When the two precision and recall indicators of label 1 simultaneously reach a good level, the accuracy of the overall model will be improved.

In the classification problem, there will be a trade-off between precision and recall. In this review, we found that recall is considered the most important, so the models whose forecast results with a recall of label 1 of 60% or more will be preferred. The precision label 1 is also of interest, so to improve this indicator, it is necessary to make a trade-off from recall, but according to the initial orientation, the recall is a top priority, so we decided to choose the 60% as a landmark that the recall label 1 that the model needs to achieve.

2.2 Model performance:

2.2.1 On the provided train dataset:

- For the dataset provided, most of the models that we build do not meet the initial criteria and orientation. The common point of the forecast results from these models is that the forecast of most transactions is not fraudulent (label 0), the forecast performance of label 1 is very poor.
- We decided to use the upsample method to increase the size of the dataset while allowing the number of label 0 and label 1 observations to become equal.

```
The shape of data after up-sampled:
Shape of class 0: (25146, 46)
Shape of class 1: (25146, 46)
```

Figure 2: Number of observations per label in the dataset after upsample.

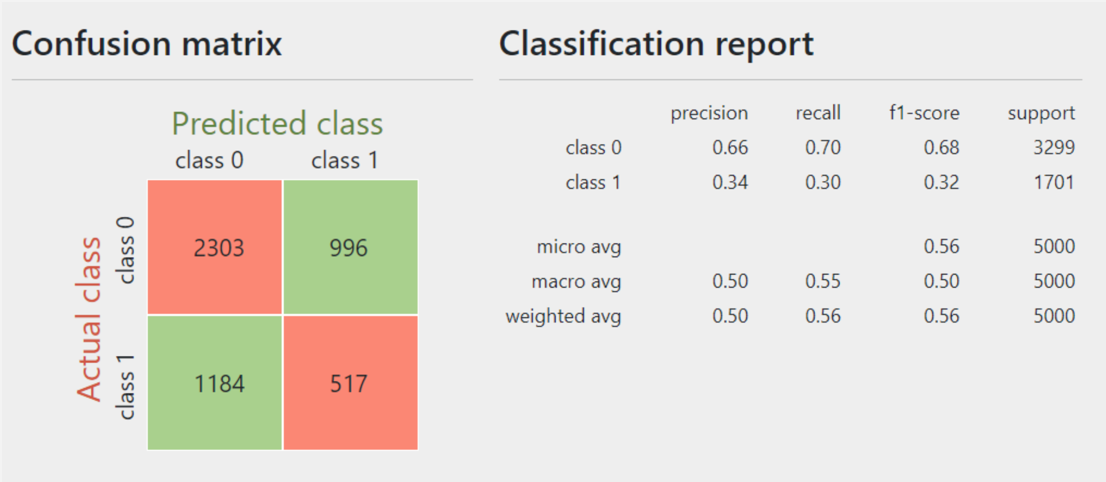
- After redeploying the models under the new dataset, Random Forest is the only model that produces predictive results that meet the criteria and directions set by the team.

```
              precision    recall  f1-score   support

           0       0.70      0.40      0.51      6335
           1       0.37      0.68      0.47      3271

    accuracy                           0.49      9606
   macro avg       0.53      0.54      0.49      9606
weighted avg       0.59      0.49      0.49      9606

Random Forest Accuracy: 49.0%
```

	Predict 0	Predict 1
Actual 0	2503	3832
Actual 1	1063	2208

Figure 3: The forecast result of the Random Forest model on the train data set has been upsample.

2.2.2 On the test dataset:



	precision	recall	f1-score	support
class 0	0.66	0.70	0.68	3299
class 1	0.34	0.30	0.32	1701
micro avg			0.56	5000
macro avg	0.50	0.55	0.50	5000
weighted avg	0.50	0.56	0.56	5000

Figure 4: The forecast results of the Random Forest model on the test set.

- The team's model correctly identified 1083 fraudulent transactions out of a total of 1701 actually fraudulent transactions (accounting for 63.7%). The recall of label 1 reached 64%, meeting the requirements of the research team.
- However, the precision of label 1 is not significantly improved compared to 34.02%, which is an insurmountable defect of the model.
- Besides, the accuracy and F1 score are not too impressive.
- However, after testing and comparing through many different machine learning models as well as through many data preprocessing methods. The Random Forest model has produced the most satisfactory results as well as the closest to the goal of the group.
- The advantages and disadvantages of the test set forecast results are similar to the advantages and disadvantages of the forecast results on the train set.

3. Direction to improve forecasting performance

- Regarding the direction of the problem: The precision of label 1 needs to be improved compared to 34.02%, the process of implementing different models found that to keep a good recall level, the precision is difficult to reach above 40%. This is something that needs to be improved to avoid resource loss as well as minimize the risk of losing customers.
- Technical issues: It is necessary to invest more in the process of researching similar issues to have the best data processing measures. In addition, algorithms and models also need to be upgraded to handle complex datasets such as the one provided.

4. Conclusions:

The model as well as the prediction results have met the initial orientations and criteria set forth. The model we built can be applied to other fields with some problems such as: detecting the possibility of cancer in patients, detecting businesses that are likely to be financially distressed, etc.