Project_02

August 15, 2021

1 Survey of the reliablity of the bank clients

Credit department of the bank requested to analyse does the marital status and quantity of child are influence on the payment of oustandins fees in the specified duration in credit contract. Incoming data - statistic with credit score of the cilents.

The result of the survey will be used for the model of evalution of **credit score** - system wich evaluate the capacity of the client to pay the oustandins fees in the specified duration in credit contract

1.1 Step 1. Open the file and conduct the EDA

[1]:
```
# import of libraries
import pandas as pd
from pymystem3 import Mystem
```

[2]:
```
# read the data and asign it to table variable
table = pd.read_csv('/datasets/data.csv')

# print table info and first 10 rows
table.info()
table.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   children          21525 non-null  int64
 1   days_employed     19351 non-null  float64
 2   dob_years         21525 non-null  int64
 3   education         21525 non-null  object
 4   education_id      21525 non-null  int64
 5   family_status     21525 non-null  object
 6   family_status_id  21525 non-null  int64
 7   gender            21525 non-null  object
 8   income_type       21525 non-null  object
 9   debt              21525 non-null  int64
 10  total_income      19351 non-null  float64
```

```
 11  purpose          21525 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

[2]:
```
   children  days_employed  dob_years education  education_id  \
0         1   -8437.673028         42                         0
1         1   -4024.803754         36                         1
2         0   -5623.422610         33                         1
3         3   -4124.747207         32                         1
4         0  340266.072047         53                         1
5         0    -926.185831         27                         0
6         0   -2879.202052         43                         0
7         0    -152.779569         50                         1
8         2   -6929.865299         35                         0
9         0   -2188.756445         41                         1

      family_status  family_status_id gender income_type  debt   total_income  \
0       /                          0      F             0  253875.639453
1       /                          0      F             0  112080.014102
2       /                          0      M             0  145885.952297
3       /                          0      M             0  267628.550329
4                                1      F             0  158616.077870
5                                1      M             0  255763.565419
6       /                          0      F             0  240525.971920
7       /                          0      M             0  135823.934197
8                                1      F             0   95856.832424
9       /                          0      M             0  144425.938277

                      purpose
0
1
2
3
4
5
6
7
8
9
```

Conclusion

1) The table has 21525 rows   12 columns.
2) In columns days_employed and total_income there are 2174 nulls, cleints without informtion on their income and occupation status. The total quanity of nuls almost 10% and sufficient for the dataset and overal statistic, therefore these data shall not be deleted.
3) In columns education there is difference in applying of register, these data shall be processed to have the nuqie formatting.

4) In columns days_employed and total_income data shall be procesed to get the understandable format for the further work and analysis.

1.2 Step 2. Data Preparation

1.2.1 Nulls processing

For the purpose of avoiding of the data loss, the nulls value the columns total_income and days_employed to be filled with mean value.

[3]:
```
# fill the nulls values in column days_employed with mean
table['days_employed'] = table['days_employed'].fillna(table['days_employed'].
 ↪mean())

#  reformatting the education column to have the lower cases
table['education'] = table['education'].str.lower()

#  decalre the function for the definition of age category
def age (age_value):
    if age_value < 35:
        return '     '
    elif 35<= age_value <=55:
        return '         '
    elif age_value>55:
        return '          '

# add the age category column to dataset
table['age_category'] = table['dob_years'].apply(age)

#  creating the datset with unique columns of age categories, education and␣
 ↪type of income
all_unique = {'education': table['education'],'income_type':
 ↪table['income_type'],'age_category': table['age_category']}
all_unique = pd.DataFrame(all_unique)
all_unique = all_unique.drop_duplicates().reset_index(drop=True)

#  declare a function for defenition of mean value for every row in table
def median_calc (row):
    B = round(table[(table['age_category'] == row['age_category']) &␣
 ↪(table['income_type'] == row['income_type']) & (table['education'] ==␣
 ↪row['education'] )].total_income.median(),2)
    return B

# add the mean value to the table
all_unique['median'] = all_unique.apply(median_calc,axis=1)

# filling the nulls with mean value
```

```python
all_unique['median'] = all_unique['median'].fillna(all_unique['median'].
 ↪median())

# display the table
display(all_unique)
```

```
/opt/conda/lib/python3.9/site-packages/numpy/lib/nanfunctions.py:1117:
RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

	education	income_type	age_category	median
0				171441.00
1				138435.87
2				131710.94
3				117894.96
4				191159.14
5				215277.93
6				114200.79
7				161129.20
8				200807.92
9				157505.97
10				134526.70
11				125560.37
12				162140.52
13				154081.92
14				145919.11
15				215453.89
16				160242.95
17				164098.31
18				175589.24
19				160298.23
20				142307.34
21				97620.69
22				140785.72
23				102598.65
24				159994.53
25				147036.63
26				173090.07
27				96330.47
28				135753.54
29				125994.91
30				132294.64
31				127021.00
32				213211.87
33				111701.53
34				184722.74
35				184062.64
36				150857.39

```
37                                        229339.20
38                                            190912.18
39                                             168979.94
40                                        127205.09
41                                          177088.85
42                                                 59956.99
43                                           157259.90
44                                                  183556.36
45                                            79432.97
46                                           155670.91
47                                          268411.21
48                                               191021.14
49                                                   98201.63
50                                          111392.23
51                                              202722.51
52                                             96989.66
53                                              499163.14
54                                               53829.13
```

[4]:
```
# creating a loop for the filling of nulls value of row total_income in table
for ind in all_unique.index:
    (table[(table['age_category']== all_unique['age_category'][ind])
           & (table['education']== all_unique['education'][ind])
           & (table['income_type']== all_unique['income_type'][ind])]) =(
        table[(table['age_category']== all_unique['age_category'][ind])
           & (table['education']== all_unique['education'][ind])
           & (table['income_type']== all_unique['income_type'][ind])]).
 ↪fillna(all_unique['median'][ind])

# display the info on table dataset
table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   children          21525 non-null  int64
 1   days_employed     21525 non-null  float64
 2   dob_years         21525 non-null  int64
 3   education         21525 non-null  object
 4   education_id      21525 non-null  int64
 5   family_status     21525 non-null  object
 6   family_status_id  21525 non-null  int64
 7   gender            21525 non-null  object
 8   income_type       21525 non-null  object
 9   debt              21525 non-null  int64
 10  total_income      21525 non-null  float64
```

```
 11   purpose            21525 non-null   object
 12   age_category       21525 non-null   object
dtypes: float64(2), int64(5), object(6)
memory usage: 2.1+ MB
```

Conclusion

All the nulls value were fullfille the further work could be started.

The nulls value culd be in data set due to te absence of provided information from clients or income equeal to zero.

1.2.2 Data type update

[5]:
```python
# changing of data type of columns total_income and days_employed to int
table['total_income'] = table['total_income'].astype('int')
table['days_employed'] = table['days_employed'].astype('int')

# display the results
table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   children          21525 non-null   int64
 1   days_employed     21525 non-null   int64
 2   dob_years         21525 non-null   int64
 3   education         21525 non-null   object
 4   education_id      21525 non-null   int64
 5   family_status     21525 non-null   object
 6   family_status_id  21525 non-null   int64
 7   gender            21525 non-null   object
 8   income_type       21525 non-null   object
 9   debt              21525 non-null   int64
 10  total_income      21525 non-null   int64
 11  purpose           21525 non-null   object
 12  age_category      21525 non-null   object
dtypes: int64(7), object(6)
memory usage: 2.1+ MB
```

Conclusion

After the changing of the data types it would easier to work with it.

Based on the latest infomration the float data were changed succesfully to int.

For the replace of the data astype method was applied due to the fact that original data were float type and there was no any reason to get the information on the errors during the data type changing.

1.2.3 Duplicates processing

[6]:
```
# searhof duplicates
print('quantity of duplicates: ',table.duplicated().sum(),'\n')

# deletion of duplicates and reset of index
table = table.drop_duplicates().reset_index(drop=True)

# check of the result
table.info()
```

```
quantity of duplicates:  71

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21454 entries, 0 to 21453
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   children          21454 non-null  int64
 1   days_employed     21454 non-null  int64
 2   dob_years         21454 non-null  int64
 3   education         21454 non-null  object
 4   education_id      21454 non-null  int64
 5   family_status     21454 non-null  object
 6   family_status_id  21454 non-null  int64
 7   gender            21454 non-null  object
 8   income_type       21454 non-null  object
 9   debt              21454 non-null  int64
 10  total_income      21454 non-null  int64
 11  purpose           21454 non-null  object
 12  age_category      21454 non-null  object
dtypes: int64(7), object(6)
memory usage: 2.1+ MB
```

Conclusion

The deletion of duplicates was made using method drop_duplicates w/o specification of exact columns for the deletion of duplicates in all rows, additionaly indexes were reseted.

Duplicates most likely appeared in dataset due to the mistakes during the creating a new records in table - human factor.

1.2.4 Lemmatization

[7]:
```
# assigning of Mystem() to variable 'm'
m = Mystem()

# lemmatization of row purpose and add lemmatized text as new row to dataset
table['purpose_category'] = table['purpose'].apply(lambda x: m.lemmatize(x))
```

```
[8]: #     Creation of two varaeblies. first - values of column purpose_category,␣
     ↪second - to save the unique value of column
     var_one = table['purpose_category']
     lem_unique = []
     for i in var_one:
         if i not in lem_unique:
             lem_unique.append(i)
     print(lem_unique)
```

```
[['    ', ' ', '   ', '\n'], ['       ', ' ', '      ', '\n'],
['        ', ' ', '      ', '\n'], ['    ', ' ', '    ', '\n'],
['    ', ' ', ' ', ' ', '   ', '\n'], ['      ', '\n'], [' ', ' ',
'     ', ' ', '    ', '\n'], ['    ', ' ', '   ', ' ', ' ', ' ',
'   ', '\n'], ['    ', ' ', '       ', '\n'], ['    ', ' ',
'       ', ' ', '       ', '\n'], ['    ', ' ', '   ', ' ',
'       ', '\n'], ['       ', ' ', '      ', ' ',
'       ', '\n'], ['       ', '\n'], ['       ', ' ',
'       ', '\n'], [' ', ' ', '    ', ' ', '      ', ' ',
'     ', '\n'], [' ', ' ', '    ', ' ', '   ', ' ', '      ',
'\n'], ['     ', ' ', ' ', ' ', '        ', ' ', '       ', '\n'],
['        ', ' ', '   ', ' ', '       ', '\n'], ['   ', '\n'],
['    ', ' ', ' ', ' ', '   ', ' ', '       ', '\n'], ['      ',
'\n'], ['      ', ' ', '       ', '\n'], ['    ', ' ', ' ', ' ',
'      ', ' ', '       ', '\n'], ['     ', ' ', '        ', ' ',
'     ', '\n'], ['   ', '\n'], ['     ', ' ', '        ', ' ',
' ', ' ', '       ', '\n'], ['    ', ' ', '   ', ' ', '   ', '\n'], ['
', '      ', '\n'], ['      ', ' ', '       ', '\n'], ['    ', '
', '       ', '\n'], ['    ', ' ', '   ', ' ', '  ', ' ', '   ',
'\n'], [' ', ' ', '    ', ' ', '      ', '\n'], ['   ', ' ', '   ',
'\n'], ['      ', ' ', '    ', ' ', '      ', '\n']]
```

Conclusion

Based on the lemmatization we can get 4 main lemms: 1)   2)   3)   4)

1.2.5 Data categorization

```
[9]: # declaration of function for categorization of data in columns␣
     ↪purpose_category by 4 main categories
     def category (input_value):
         if '    ' in input_value:
             return '     '
         elif '      ' in input_value:
             return '      '
         elif '       ' in input_value:
             return '       '
```

```
    else:
        return '        '

# declaration of function for categorization by presence/absence of debt
def credit_debt (input_value):
    if input_value == 0:
        return '      '
    else:
        return '        '

# declaration of function for categorization by presence/absence of child
def kids (input_value):
    if input_value == 0:
        return '      '
    else:
        return '      '

# apply of declared fucntion
table['purpose_category'] = table['purpose_category'].apply(category)
table['debt_status'] = table['debt'].apply(credit_debt)
table['kids_status'] = table['children'].apply(kids)

# chechk of the results
table.head(10)
```

[9]:

```
   children  days_employed  dob_years education  education_id  \
0         1          -8437         42                       0
1         1          -4024         36                      1
2         0          -5623         33                      1
3         3          -4124         32                      1
4         0         340266         53                      1
5         0           -926         27                       0
6         0          -2879         43                       0
7         0           -152         50                      1
8         2          -6929         35                       0
9         0          -2188         41                      1

      family_status  family_status_id gender  income_type  debt  total_income  \
0       /                     0       F              0       253875
1       /                     0       F              0       112080
2       /                     0       M              0       145885
3       /                     0       M              0       267628
4                           1       F              0       158616
5                           1       M              0       255763
6       /                     0       F              0       240525
7       /                     0       M              0       135823
8                           1       F              0        95856
```

```
9       /                    0     M            0       144425

                       purpose     age_category purpose_category debt_status  \
0
1
2
3
4
5
6
7
8
9

  kids_status
0
1
2
3
4
5
6
7
8
9
```

Conslusion

For the completion of catogerization by column purpose_category were selected 4 main lemms because one of such values is in every row of this column.

For other categorization were selected absence/presence of child and absence/presence debt for the further data analysis and answering on the questions.

1.3 Step 3. Question answers

Is there a dependency between the presence of a child and payment of outstanding fees in the specified duration in the credit contract?

[10]:
```
# creation of a new dataset equal to "table"
kids_kredit_check = table.copy()

# creation of new columns with status of presence of the child and debt and␣
↪group by it's values
kids_kredit_check['category_kids_kredit'] = kids_kredit_check['kids_status']+',␣
↪'+kids_kredit_check['debt_status']
kids_kredit_check = kids_kredit_check.groupby('category_kids_kredit')['debt'].
↪count()
```

```
# display the results
display(kids_kredit_check)

# checking that all data is included
print(kids_kredit_check.sum()==table['debt'].count(),'\n')

# calculation of percentages by each category
precent_withno_kids = kids_kredit_check.iloc[1]/(kids_kredit_check.
 ↪iloc[0]+kids_kredit_check.iloc[1])
precent_with_kids = kids_kredit_check.iloc[3]/(kids_kredit_check.
 ↪iloc[2]+kids_kredit_check.iloc[3])

# display of the results
print('              ,            : {:.2%}'.format(precent_with_kids))
print('               ,            : {:.2%}'.format(precent_withno_kids))
```

```
category_kids_kredit
     ,             13028
     ,             1063
     ,              6685
     ,             678
Name: debt, dtype: int64
```

```
True
```

```
               ,           : 9.21%
                ,           : 7.54%
```

Conclusion

- Based on the information from bank there is a dependency: client with a child on 1.7% frequently will not pay the outstanding fees in the specified duration in the credit contract.
- However the data has only 21454 records, most likely the sufficient increase of the records could influnce on the result of the statistic. _____

Is there a dependency on the marital status and payment of outstanding fees in the specified duration in the credit contract?

[11]:
```
# creation of new columns with marital status and debt
table['marriage_debt_status'] = table['family_status']+', '+table['debt_status']

# group by column marriage_debt_status
marriage_check = table.groupby('marriage_debt_status')['debt'].count()

# display the results
display(marriage_check)

# checking that all data is included
print(marriage_check.sum()==table['debt'].count(),'\n')
```

```
# calculation of percentages by each category
precent_not_married = marriage_check.iloc[1]/(marriage_check.
↪iloc[0]+marriage_check.iloc[1])
precent_divorced = marriage_check.iloc[3]/(marriage_check.
↪iloc[2]+marriage_check.iloc[3])
precent_widow = marriage_check.iloc[5]/(marriage_check.iloc[4]+marriage_check.
↪iloc[5])
precent_civil_partners = marriage_check.iloc[7]/(marriage_check.
↪iloc[6]+marriage_check.iloc[7])
precent_married = marriage_check.iloc[9]/(marriage_check.iloc[8]+marriage_check.
↪iloc[9])

# display of the results
print('              ,            : {:.2%}'.format(precent_married))
print('          ,                              : {:.2%}'.
↪format(precent_civil_partners))
print('               ,            : {:.2%}'.format(precent_divorced))
print('              ,            : {:.2%}'.format(precent_widow))
print('         ,                           : {:.2%}'.format(precent_not_married))
```

```
marriage_debt_status
      /       ,              2536
      /       ,              274
     ,                           1110
     ,                            85
    /    ,                       896
    /    ,                       63
         ,                   3763
         ,                   388
   /     ,                    11408
   /     ,                     931
Name: debt, dtype: int64
```

```
True
```

```
              ,            : 7.55%
         ,                              : 9.35%
               ,            : 7.11%
              ,            : 6.57%
         ,                           : 9.75%
```

Conclusion

- The lowest percentage of clients with debt overdue - it's widowed clients- 6,5%.
- Than divorsed cliens - 7,1%.
- Only 7.55% of clients who married have debt verdue.
- the highest percentage of debt overdue have clients who is not married or informal married - 9,75% and 9,35%

Is there a dependency on the income grade and payment of outstanding fees in the specified duration in the credit contract?

[12]:
```python
# calculation of average income and categorization of clients
range_groups = pd.qcut(table['total_income'],q=3)
range_groups = range_groups.drop_duplicates().sort_values().reset_index(drop=1)
print(range_groups)

# function declare for definition of income grade
def income (income_value):
    if income_value in range_groups[0]:
        return '           '
    elif income_value in range_groups[1]:
        return '        '
    else:
        return '           '

# add a column with income grade
table['income_debt_status'] = table['total_income'].apply(income)

#  calculation of percentages by each category
table.groupby('income_debt_status')['debt'].agg(['count', 'sum', lambda x: '{:.
 ↪2%} '.format(x.mean())])
```

```
0    (20666.999, 119869.0]
1     (119869.0, 173597.0]
2    (173597.0, 2265604.0]
Name: total_income, dtype: category
Categories (3, interval[float64]): [(20666.999, 119869.0] < (119869.0, 173597.0]
< (173597.0, 2265604.0]]
```

[12]:
```
                     count  sum <lambda_0>
income_debt_status
                     7151   526      7.36%
                     7152   580      8.11%
                       7151   635      8.88%
```

Conclusion

- Clients with average grade income are more often will not pay the outstanding fees in the specified duration in the credit contract (8.88%).
- Clients with high income grade - has the smallest precentage of debtors that exceed the credit time limit 7%.
- Amongst of clients with lower income grade - 8% of such debtors _____

How does the different credite purpose influence on the payment of the outstanding fees in the specified duration in the credit contract?

```
[13]: # add a new column purpose_debt_status (concat of debt_status and purpose␣
     ↪category)
     table['purpose_debt_status'] = table['purpose_category']+',␣
     ↪'+table['debt_status']

     # group by new column
     purpose_check = table.groupby('purpose_debt_status')['debt'].count()

     # display the result
     display(purpose_check)

     # check that all data is included
     print(purpose_check.sum()==table['debt'].count(),'\n')

     # calculation of percentages by each category
     precent_auto = purpose_check.iloc[1]/(purpose_check.iloc[0]+purpose_check.
     ↪iloc[1])
     precent_realty = purpose_check.iloc[3]/(purpose_check.iloc[2]+purpose_check.
     ↪iloc[3])
     precent_education_= purpose_check.iloc[5]/(purpose_check.iloc[4]+purpose_check.
     ↪iloc[5])
     precent_marriage = purpose_check.iloc[7]/(purpose_check.iloc[6]+purpose_check.
     ↪iloc[7])

     # display of the resutlts
     print('                          ,            : {:.2%}'.format(precent_auto))
     print('                           ,            : {:.2%}'.format(precent_realty))
     print('                          ,            : {:.2%}'.
     ↪format(precent_education_))
     print('                        ,            : {:.2%}'.format(precent_marriage))
```

```
purpose_debt_status
     ,                3903
     ,                403
      ,              10029
      ,               782
     ,                3643
     ,                370
   ,                  2138
   ,                  186
Name: debt, dtype: int64

True

                          ,            : 9.36%
                           ,            : 7.23%
                          ,            : 9.22%
                        ,            : 8.00%
```

Conclusion

- Only 7% of clients who got credit on purchase of real estate has debt overdue.
- 8% of clients who got credit on wedding has debt overdue.
- The gighest percentaage of clients who has debt overdue got credit on auto or education (9,3% 9,2%)

1.4 Step 4. General conclusion

During data analysis the following infomration was obtained:

1.5 ***Most relaible clients:***

- Clients who's got credit on the purchase of real estate (only 7.23% debt overdue)
- Clients with marital status - widow / er(6.57% debt overdue)
- Clinets w/o kids (7.54% debt overdue)
- Clients with income bove average (7.36% debt overdue)

Unrelaible clients: - Clients who's got credit on purchasing of auto (9.36% debt overdue) - Clients who were not married (9.75% debt overdue) - Clients with child (9.21% debt overdue) - Clients with average grade of income (8.88% debt overdue)