draft

September 27, 2024

[1]:
```python
# Importing necessary libraries
import pandas as pd

# Load application data
train_data = pd.read_csv('/home/ignatiusvmk/Downloads/home-credit-default-risk/
 ↪application_train.csv')
test_data = pd.read_csv('/home/ignatiusvmk/Downloads/home-credit-default-risk/
 ↪application_test.csv')

# Display a sample of the data
print(train_data.head(5))

# Checking the target variable distribution
train_data['TARGET'].value_counts(normalize=True)

#/home/ignatiusvmk/Downloads/home-credit-default-risk/
```

```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0      100002       1         Cash loans           M            N
1      100003       0         Cash loans           F            N
2      100004       0    Revolving loans           M            Y
3      100006       0         Cash loans           F            N
4      100007       0         Cash loans           M            N

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
0               Y             0          202500.0    406597.5      24700.5
1               N             0          270000.0   1293502.5      35698.5
2               Y             0           67500.0    135000.0       6750.0
3               Y             0          135000.0    312682.5      29686.5
4               Y             0          121500.0    513000.0      21865.5

   …  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
0  …                0                0                0                0
1  …                0                0                0                0
2  …                0                0                0                0
3  …                0                0                0                0
4  …                0                0                0                0
```

```
  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                        0.0                       0.0
1                        0.0                       0.0
2                        0.0                       0.0
3                        NaN                       NaN
4                        0.0                       0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0
1                         0.0                        0.0
2                         0.0                        0.0
3                         NaN                        NaN
4                         0.0                        0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0

[5 rows x 122 columns]
```

[1]:
```
TARGET
0    0.919271
1    0.080729
Name: proportion, dtype: float64
```

[2]:
```
train_data.head()
```

[2]:
```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0      100002       1         Cash loans           M            N
1      100003       0         Cash loans           F            N
2      100004       0    Revolving loans           M            Y
3      100006       0         Cash loans           F            N
4      100007       0         Cash loans           M            N

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
0               Y             0          202500.0    406597.5      24700.5
1               N             0          270000.0   1293502.5      35698.5
2               Y             0           67500.0    135000.0       6750.0
3               Y             0          135000.0    312682.5      29686.5
4               Y             0          121500.0    513000.0      21865.5

   …  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
0  …                0                0                0                0
1  …                0                0                0                0
2  …                0                0                0                0
```

```
3  …                    0                 0                 0                 0
4  …                    0                 0                 0                 0

  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                        0.0                        0.0
1                        0.0                        0.0
2                        0.0                        0.0
3                        NaN                        NaN
4                        0.0                        0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0
1                         0.0                        0.0
2                         0.0                        0.0
3                         NaN                        NaN
4                         0.0                        0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0

[5 rows x 122 columns]
```

[3]:
```python
train_data.shape
```

[3]:
```
(307511, 122)
```

[4]:
```python
cols_to_drop = [
    'FLAG_MOBIL', # Low variance
    'FONDKAPREMONT_MODE', 'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE',␣
 ↪'EMERGENCYSTATE_MODE',
    'APARTMENTS_MODE', 'APARTMENTS_MEDI', 'APARTMENTS_AVG',
    'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5',
    'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9',
    'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12',␣
 ↪'FLAG_DOCUMENT_13',
    'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16',␣
 ↪'FLAG_DOCUMENT_17',
    'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',␣
 ↪'FLAG_DOCUMENT_21',
    'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',␣
 ↪'AMT_REQ_CREDIT_BUREAU_WEEK',
    'AMT_REQ_CREDIT_BUREAU_MON', 'WALLSMATERIAL_MODE'
```

```
    # Add any other columns with missing data or low variance after checking␣
 ↪them.
]
train_data.drop(columns=cols_to_drop, inplace=True)
```

[5]:
```
train_data.shape
```

[5]:
```
(307511, 90)
```

[6]:
```
# Load bureau data
bureau = pd.read_csv('/home/ignatiusvmk/Downloads/home-credit-default-risk/
 ↪bureau.csv')

bureau.head()
```

[6]:
```
   SK_ID_CURR  SK_ID_BUREAU CREDIT_ACTIVE CREDIT_CURRENCY  DAYS_CREDIT  \
0      215354       5714462        Closed      currency 1         -497
1      215354       5714463        Active      currency 1         -208
2      215354       5714464        Active      currency 1         -203
3      215354       5714465        Active      currency 1         -203
4      215354       5714466        Active      currency 1         -629

   CREDIT_DAY_OVERDUE  DAYS_CREDIT_ENDDATE  DAYS_ENDDATE_FACT  \
0                   0               -153.0             -153.0
1                   0               1075.0                NaN
2                   0                528.0                NaN
3                   0                  NaN                NaN
4                   0               1197.0                NaN

   AMT_CREDIT_MAX_OVERDUE  CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  \
0                     NaN                   0         91323.0
1                     NaN                   0        225000.0
2                     NaN                   0        464323.5
3                     NaN                   0         90000.0
4                 77674.5                   0       2700000.0

   AMT_CREDIT_SUM_DEBT  AMT_CREDIT_SUM_LIMIT  AMT_CREDIT_SUM_OVERDUE  \
0                  0.0                   NaN                     0.0
1             171342.0                   NaN                     0.0
2                  NaN                   NaN                     0.0
3                  NaN                   NaN                     0.0
4                  NaN                   NaN                     0.0

       CREDIT_TYPE  DAYS_CREDIT_UPDATE  AMT_ANNUITY
0  Consumer credit                -131          NaN
1      Credit card                 -20          NaN
2  Consumer credit                 -16          NaN
```

```
3      Credit card                        -16          NaN
4  Consumer credit                        -21          NaN
```

[7]:
```
bureau.shape
```

[7]:
```
(1716428, 17)
```

[8]:
```
# Aggregation example: Sum and mean of credit amount
bureau_agg = bureau.groupby('SK_ID_CURR').agg({
    'AMT_CREDIT_SUM': ['sum', 'mean'],
    'CREDIT_DAY_OVERDUE': ['max'],
    'DAYS_CREDIT': ['mean'],
}).reset_index()

bureau_agg.head()
```

[8]:
```
  SK_ID_CURR AMT_CREDIT_SUM                CREDIT_DAY_OVERDUE  DAYS_CREDIT
                        sum            mean               max         mean
0     100001    1453365.000  207623.571429                  0  -735.000000
1     100002     865055.565  108131.945625                  0  -874.000000
2     100003    1017400.500  254350.125000                  0 -1400.750000
3     100004     189037.800   94518.900000                  0  -867.000000
4     100005     657126.000  219042.000000                  0  -190.666667
```

[9]:
```
# Rename the columns
bureau_agg.columns = ['SK_ID_CURR', 'CREDIT_SUM_TOTAL', 'CREDIT_SUM_MEAN',␣
 ↪'CREDIT_OVERDUE_MAX', 'CREDIT_DURATION_MEAN']
bureau_agg.head()
```

[9]:
```
   SK_ID_CURR  CREDIT_SUM_TOTAL  CREDIT_SUM_MEAN  CREDIT_OVERDUE_MAX  \
0      100001       1453365.000    207623.571429                   0
1      100002        865055.565    108131.945625                   0
2      100003       1017400.500    254350.125000                   0
3      100004        189037.800     94518.900000                   0
4      100005        657126.000    219042.000000                   0

   CREDIT_DURATION_MEAN
0           -735.000000
1           -874.000000
2          -1400.750000
3           -867.000000
4           -190.666667
```

[10]:
```
# Merge with train data
train_data = train_data.merge(bureau_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(bureau_agg, on='SK_ID_CURR', how='left')
```

[11]:
```
train_data.head()
```

[11]:
```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0      100002       1         Cash loans           M            N
1      100003       0         Cash loans           F            N
2      100004       0    Revolving loans           M            Y
3      100006       0         Cash loans           F            N
4      100007       0         Cash loans           M            N

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
0               Y             0          202500.0    406597.5      24700.5
1               N             0          270000.0   1293502.5      35698.5
2               Y             0           67500.0    135000.0       6750.0
3               Y             0          135000.0    312682.5      29686.5
4               Y             0          121500.0    513000.0      21865.5

   …  DEF_30_CNT_SOCIAL_CIRCLE OBS_60_CNT_SOCIAL_CIRCLE  \
0  …                       2.0                      2.0
1  …                       0.0                      1.0
2  …                       0.0                      0.0
3  …                       0.0                      2.0
4  …                       0.0                      0.0

  DEF_60_CNT_SOCIAL_CIRCLE DAYS_LAST_PHONE_CHANGE AMT_REQ_CREDIT_BUREAU_QRT  \
0                      2.0                -1134.0                       0.0
1                      0.0                 -828.0                       0.0
2                      0.0                 -815.0                       0.0
3                      0.0                 -617.0                       NaN
4                      0.0                -1106.0                       0.0

  AMT_REQ_CREDIT_BUREAU_YEAR  CREDIT_SUM_TOTAL  CREDIT_SUM_MEAN  \
0                        1.0        865055.565    108131.945625
1                        0.0       1017400.500    254350.125000
2                        0.0        189037.800     94518.900000
3                        NaN               NaN              NaN
4                        0.0        146250.000    146250.000000

   CREDIT_OVERDUE_MAX  CREDIT_DURATION_MEAN
0                 0.0               -874.00
1                 0.0              -1400.75
2                 0.0               -867.00
3                 NaN                   NaN
4                 0.0              -1149.00

[5 rows x 94 columns]
```

[12]:
```python
bureau_agg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 305811 entries, 0 to 305810
```

```
Data columns (total 5 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   SK_ID_CURR            305811 non-null  int64
 1   CREDIT_SUM_TOTAL      305811 non-null  float64
 2   CREDIT_SUM_MEAN       305809 non-null  float64
 3   CREDIT_OVERDUE_MAX    305811 non-null  int64
 4   CREDIT_DURATION_MEAN  305811 non-null  float64
dtypes: float64(3), int64(2)
memory usage: 11.7 MB
```

[13]:
```
bureau_agg.head()
```

[13]:
```
   SK_ID_CURR  CREDIT_SUM_TOTAL  CREDIT_SUM_MEAN  CREDIT_OVERDUE_MAX  \
0      100001       1453365.000    207623.571429                   0
1      100002        865055.565    108131.945625                   0
2      100003       1017400.500    254350.125000                   0
3      100004        189037.800     94518.900000                   0
4      100005        657126.000    219042.000000                   0

   CREDIT_DURATION_MEAN
0           -735.000000
1           -874.000000
2          -1400.750000
3           -867.000000
4           -190.666667
```

[14]:
```
test_data.head()
```

[14]:
```
   SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
0      100001         Cash loans           F            N               Y
1      100005         Cash loans           M            N               Y
2      100013         Cash loans           M            Y               Y
3      100028         Cash loans           F            N               Y
4      100038         Cash loans           M            Y               N

   CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  AMT_GOODS_PRICE  \
0             0          135000.0    568800.0      20560.5         450000.0
1             0           99000.0    222768.0      17370.0         180000.0
2             0          202500.0    663264.0      69777.0         630000.0
3             2          315000.0   1575000.0      49018.5        1575000.0
4             1          180000.0    625500.0      32067.0         625500.0

   … AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0  …                        0.0                       0.0
1  …                        0.0                       0.0
2  …                        0.0                       0.0
3  …                        0.0                       0.0
```

```
4  …                         NaN                         NaN

  AMT_REQ_CREDIT_BUREAU_WEEK AMT_REQ_CREDIT_BUREAU_MON  \
0                        0.0                        0.0
1                        0.0                        0.0
2                        0.0                        0.0
3                        0.0                        0.0
4                        NaN                        NaN

  AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR  CREDIT_SUM_TOTAL  \
0                       0.0                         0.0        1453365.00
1                       0.0                         3.0         657126.00
2                       1.0                         4.0        2072280.06
3                       0.0                         3.0        1520875.08
4                       NaN                         NaN               NaN

   CREDIT_SUM_MEAN  CREDIT_OVERDUE_MAX  CREDIT_DURATION_MEAN
0    207623.571429                 0.0           -735.000000
1    219042.000000                 0.0           -190.666667
2    518070.015000                 0.0          -1737.500000
3    126739.590000                 0.0          -1401.750000
4              NaN                 NaN                   NaN

[5 rows x 125 columns]
```

[15]:
```python
# Load bureau balance data
bureau_balance = pd.read_csv('/home/ignatiusvmk/Downloads/
 ↪home-credit-default-risk/bureau_balance.csv')

# Example: Count of months with a missed payment

# Replace categorical values with numeric ones
bureau_balance['STATUS'] = bureau_balance['STATUS'].replace(['C', '0'], 0).
 ↪replace(['1', '2', '3', '4', '5'], 1)

# Convert the STATUS column to numeric to handle any unexpected non-numeric␣
 ↪values
bureau_balance['STATUS'] = pd.to_numeric(bureau_balance['STATUS'],␣
 ↪errors='coerce')

# Group by SK_ID_BUREAU and aggregate status as sum (missed payments) and count␣
 ↪(total months)
bureau_balance_agg = bureau_balance.groupby('SK_ID_BUREAU').agg({
    'STATUS': ['sum', 'count']
}).reset_index()

# Rename the columns
```

```python
bureau_balance_agg.columns = ['SK_ID_BUREAU', 'MISSED_PAYMENTS', 'TOTAL_MONTHS']

bureau_balance_agg.head()
```

[15]:

	SK_ID_BUREAU	MISSED_PAYMENTS	TOTAL_MONTHS
0	5001709	0.0	86
1	5001710	0.0	53
2	5001711	0.0	3
3	5001712	0.0	19
4	5001713	0.0	0

1. Data Sourcing which is all done
2. Data Cleaning Removing/handling null values Dropping Unneccessary columns Handle Outliers Handle categorized data (Y,N / F,M,C)
3. Data Feature Engineering Encoding, Scaling data .etc Sampling to find and Imbalanced classes
4. Choose Features
5. Training the model
6. Evaluation
7. Hyperparam tuning

[16]:
```python
train_data.head()
```

[16]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	\
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	

	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	\
0	Y	0	202500.0	406597.5	24700.5	
1	N	0	270000.0	1293502.5	35698.5	
2	Y	0	67500.0	135000.0	6750.0	
3	Y	0	135000.0	312682.5	29686.5	
4	Y	0	121500.0	513000.0	21865.5	

	…	DEF_30_CNT_SOCIAL_CIRCLE	OBS_60_CNT_SOCIAL_CIRCLE	\
0	…	2.0	2.0	
1	…	0.0	1.0	
2	…	0.0	0.0	
3	…	0.0	2.0	
4	…	0.0	0.0	

	DEF_60_CNT_SOCIAL_CIRCLE	DAYS_LAST_PHONE_CHANGE	AMT_REQ_CREDIT_BUREAU_QRT	\
0	2.0	-1134.0	0.0	

```
1                   0.0                -828.0                         0.0
2                   0.0                -815.0                         0.0
3                   0.0                -617.0                         NaN
4                   0.0               -1106.0                         0.0

  AMT_REQ_CREDIT_BUREAU_YEAR  CREDIT_SUM_TOTAL  CREDIT_SUM_MEAN  \
0                        1.0        865055.565    108131.945625
1                        0.0       1017400.500    254350.125000
2                        0.0        189037.800     94518.900000
3                        NaN               NaN              NaN
4                        0.0        146250.000    146250.000000

   CREDIT_OVERDUE_MAX  CREDIT_DURATION_MEAN
0                 0.0               -874.00
1                 0.0              -1400.75
2                 0.0               -867.00
3                 NaN                   NaN
4                 0.0              -1149.00

[5 rows x 94 columns]
```

[17]:
```
# Load POS_CASH_balance data
pos_cash_balance = pd.read_csv('/home/ignatiusvmk/Downloads/
 ↪home-credit-default-risk/POS_CASH_balance.csv')

# Aggregation example: Count of active loans
pos_cash_agg = pos_cash_balance.groupby('SK_ID_CURR').agg({
    'MONTHS_BALANCE': 'count',  # Count of months with POS loans
    'SK_DPD': ['mean', 'sum'],  # Delay in payment (mean and total)
}).reset_index()

# Rename columns
pos_cash_agg.columns = ['SK_ID_CURR', 'POS_LOANS_COUNT', 'POS_DPD_MEAN',␣
 ↪'POS_DPD_TOTAL']

# Merge with train and test data
train_data = train_data.merge(pos_cash_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(pos_cash_agg, on='SK_ID_CURR', how='left')
```

[18]:
```
train_data.head()
```

[18]:
```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0      100002       1         Cash loans           M            N
1      100003       0         Cash loans           F            N
2      100004       0    Revolving loans           M            Y
3      100006       0         Cash loans           F            N
4      100007       0         Cash loans           M            N
```

```
  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
0               Y             0          202500.0    406597.5      24700.5
1               N             0          270000.0   1293502.5      35698.5
2               Y             0           67500.0    135000.0       6750.0
3               Y             0          135000.0    312682.5      29686.5
4               Y             0          121500.0    513000.0      21865.5

   …  DAYS_LAST_PHONE_CHANGE AMT_REQ_CREDIT_BUREAU_QRT  \
0  …                 -1134.0                       0.0
1  …                  -828.0                       0.0
2  …                  -815.0                       0.0
3  …                  -617.0                       NaN
4  …                 -1106.0                       0.0

  AMT_REQ_CREDIT_BUREAU_YEAR CREDIT_SUM_TOTAL CREDIT_SUM_MEAN  \
0                        1.0       865055.565   108131.945625
1                        0.0      1017400.500   254350.125000
2                        0.0       189037.800    94518.900000
3                        NaN              NaN             NaN
4                        0.0       146250.000   146250.000000

  CREDIT_OVERDUE_MAX  CREDIT_DURATION_MEAN  POS_LOANS_COUNT  POS_DPD_MEAN  \
0                0.0               -874.00             19.0           0.0
1                0.0              -1400.75             28.0           0.0
2                0.0               -867.00              4.0           0.0
3                NaN                   NaN             21.0           0.0
4                0.0              -1149.00             66.0           0.0

   POS_DPD_TOTAL
0            0.0
1            0.0
2            0.0
3            0.0
4            0.0

[5 rows x 97 columns]
```

[19]:
```
test_data.head()
```

[19]:
```
   SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
0      100001         Cash loans           F            N               Y
1      100005         Cash loans           M            N               Y
2      100013         Cash loans           M            Y               Y
3      100028         Cash loans           F            N               Y
4      100038         Cash loans           M            Y               N
```

```
   CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  AMT_GOODS_PRICE  \
0             0          135000.0    568800.0      20560.5         450000.0
1             0           99000.0    222768.0      17370.0         180000.0
2             0          202500.0    663264.0      69777.0         630000.0
3             2          315000.0   1575000.0      49018.5        1575000.0
4             1          180000.0    625500.0      32067.0         625500.0

   … AMT_REQ_CREDIT_BUREAU_MON AMT_REQ_CREDIT_BUREAU_QRT  \
0  …                       0.0                       0.0
1  …                       0.0                       0.0
2  …                       0.0                       1.0
3  …                       0.0                       0.0
4  …                       NaN                       NaN

  AMT_REQ_CREDIT_BUREAU_YEAR CREDIT_SUM_TOTAL CREDIT_SUM_MEAN  \
0                        0.0       1453365.00   207623.571429
1                        3.0        657126.00   219042.000000
2                        4.0       2072280.06   518070.015000
3                        3.0       1520875.08   126739.590000
4                        NaN              NaN             NaN

   CREDIT_OVERDUE_MAX  CREDIT_DURATION_MEAN  POS_LOANS_COUNT  POS_DPD_MEAN  \
0                 0.0           -735.000000              9.0      0.777778
1                 0.0           -190.666667             11.0      0.000000
2                 0.0          -1737.500000             36.0      0.944444
3                 0.0          -1401.750000             31.0      0.000000
4                 NaN                   NaN             13.0      0.000000

   POS_DPD_TOTAL
0            7.0
1            0.0
2           34.0
3            0.0
4            0.0

[5 rows x 128 columns]
```

[20]:
```python
# Load credit card balance data
credit_card_balance = pd.read_csv('/home/ignatiusvmk/Downloads/
 ↪home-credit-default-risk/credit_card_balance.csv')

# Aggregation example: Average balance over time
credit_card_agg = credit_card_balance.groupby('SK_ID_CURR').agg({
    'AMT_BALANCE': ['mean', 'max'],
    'MONTHS_BALANCE': 'count'
}).reset_index()
```

```
# Rename columns
credit_card_agg.columns = ['SK_ID_CURR', 'CREDIT_BALANCE_MEAN',␣
 ↪'CREDIT_BALANCE_MAX', 'CREDIT_CARD_MONTHS']

# Merge with train and test data
train_data = train_data.merge(credit_card_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(credit_card_agg, on='SK_ID_CURR', how='left')
```

[21]:
```
print(test_data.head())
```

```
   SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
0      100001         Cash loans           F            N               Y
1      100005         Cash loans           M            N               Y
2      100013         Cash loans           M            Y               Y
3      100028         Cash loans           F            N               Y
4      100038         Cash loans           M            Y               N

   CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  AMT_GOODS_PRICE  \
0             0          135000.0    568800.0      20560.5         450000.0
1             0           99000.0    222768.0      17370.0         180000.0
2             0          202500.0    663264.0      69777.0         630000.0
3             2          315000.0   1575000.0      49018.5        1575000.0
4             1          180000.0    625500.0      32067.0         625500.0

   … CREDIT_SUM_TOTAL CREDIT_SUM_MEAN CREDIT_OVERDUE_MAX  \
0  …       1453365.00   207623.571429                0.0
1  …        657126.00   219042.000000                0.0
2  …       2072280.06   518070.015000                0.0
3  …       1520875.08   126739.590000                0.0
4  …              NaN             NaN                NaN

  CREDIT_DURATION_MEAN POS_LOANS_COUNT  POS_DPD_MEAN  POS_DPD_TOTAL  \
0          -735.000000             9.0      0.777778            7.0
1          -190.666667            11.0      0.000000            0.0
2         -1737.500000            36.0      0.944444           34.0
3         -1401.750000            31.0      0.000000            0.0
4                  NaN            13.0      0.000000            0.0

   CREDIT_BALANCE_MEAN  CREDIT_BALANCE_MAX  CREDIT_CARD_MONTHS
0                  NaN                 NaN                 NaN
1                  NaN                 NaN                 NaN
2         18159.919219          161420.220                96.0
3          8085.058163           37335.915                49.0
4                  NaN                 NaN                 NaN

[5 rows x 131 columns]
```

[22]:
```
# Load installments payments data
installments_payments = pd.read_csv('/home/ignatiusvmk/Downloads/
↪home-credit-default-risk/installments_payments.csv')

# Aggregation example: Total and mean of payments
installments_agg = installments_payments.groupby('SK_ID_CURR').agg({
    'AMT_PAYMENT': ['sum', 'mean'],
    'DAYS_INSTALMENT': 'count'
}).reset_index()

# Rename columns
installments_agg.columns = ['SK_ID_CURR', 'PAYMENT_TOTAL', 'PAYMENT_MEAN',␣
↪'TOTAL_INSTALLMENTS']

# Merge with train and test data
train_data = train_data.merge(installments_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(installments_agg, on='SK_ID_CURR', how='left')
```

[23]:
```
# Fill missing values in numeric columns with the mean
numeric_cols = train_data.select_dtypes(include=['number']).columns
train_data[numeric_cols] = train_data[numeric_cols].
↪fillna(train_data[numeric_cols].mean())
numeric_cols_test = test_data.select_dtypes(include=['number']).columns
test_data[numeric_cols_test] = test_data[numeric_cols_test].
↪fillna(test_data[numeric_cols_test].mean())

# Fill missing values in non-numeric columns (e.g., categorical) with the mode
non_numeric_cols = train_data.select_dtypes(exclude=['number']).columns
train_data[non_numeric_cols] = train_data[non_numeric_cols].
↪fillna(train_data[non_numeric_cols].mode().iloc[0])
non_numeric_cols_test = test_data.select_dtypes(exclude=['number']).columns
test_data[non_numeric_cols_test] = test_data[non_numeric_cols_test].
↪fillna(test_data[non_numeric_cols_test].mode().iloc[0])
```

[24]:
```
train_data.select_dtypes(exclude=['float'])

# train_data.info()
```

[24]:
```
        SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0           100002       1         Cash loans           M            N
1           100003       0         Cash loans           F            N
2           100004       0    Revolving loans           M            Y
3           100006       0         Cash loans           F            N
4           100007       0         Cash loans           M            N
…              …       …                 …           …            …
307506      456251       0         Cash loans           M            N
307507      456252       0         Cash loans           F            N
```

```
307508      456253       0         Cash loans          F            N
307509      456254       1         Cash loans          F            N
307510      456255       0         Cash loans          F            N

       FLAG_OWN_REALTY  CNT_CHILDREN NAME_TYPE_SUITE      NAME_INCOME_TYPE  \
0                    Y             0   Unaccompanied               Working
1                    N             0          Family         State servant
2                    Y             0   Unaccompanied               Working
3                    Y             0   Unaccompanied               Working
4                    Y             0   Unaccompanied               Working
…                  …             …              …                   …
307506               N             0   Unaccompanied               Working
307507               Y             0   Unaccompanied             Pensioner
307508               Y             0   Unaccompanied               Working
307509               Y             0   Unaccompanied  Commercial associate
307510               N             0   Unaccompanied  Commercial associate

                  NAME_EDUCATION_TYPE  … REGION_RATING_CLIENT_W_CITY  \
0        Secondary / secondary special  …                           2
1                     Higher education  …                           1
2        Secondary / secondary special  …                           2
3        Secondary / secondary special  …                           2
4        Secondary / secondary special  …                           2
…                                   …  …                         …
307506  Secondary / secondary special  …                           1
307507  Secondary / secondary special  …                           2
307508               Higher education  …                           3
307509  Secondary / secondary special  …                           2
307510               Higher education  …                           1

       WEEKDAY_APPR_PROCESS_START  HOUR_APPR_PROCESS_START  \
0                       WEDNESDAY                       10
1                          MONDAY                       11
2                          MONDAY                        9
3                       WEDNESDAY                       17
4                        THURSDAY                       11
…                             …                      …
307506                   THURSDAY                       15
307507                     MONDAY                        8
307508                   THURSDAY                        9
307509                  WEDNESDAY                        9
307510                   THURSDAY                       20

        REG_REGION_NOT_LIVE_REGION  REG_REGION_NOT_WORK_REGION  \
0                                0                           0
1                                0                           0
2                                0                           0
```

```
3                                 0                            0
4                                 0                            0
...                             ...                          ...
307506                            0                            0
307507                            0                            0
307508                            0                            0
307509                            0                            0
307510                            0                            0

        LIVE_REGION_NOT_WORK_REGION  REG_CITY_NOT_LIVE_CITY  \
0                                 0                       0
1                                 0                       0
2                                 0                       0
3                                 0                       0
4                                 0                       0
...                             ...                     ...
307506                            0                       0
307507                            0                       0
307508                            0                       0
307509                            0                       1
307510                            0                       0

        REG_CITY_NOT_WORK_CITY  LIVE_CITY_NOT_WORK_CITY  \
0                            0                        0
1                            0                        0
2                            0                        0
3                            0                        0
4                            1                        1
...                        ...                      ...
307506                       0                        0
307507                       0                        0
307508                       1                        1
307509                       1                        0
307510                       1                        1

             ORGANIZATION_TYPE
0       Business Entity Type 3
1                       School
2                   Government
3       Business Entity Type 3
4                     Religion
...                        ...
307506                Services
307507                     XNA
307508                  School
307509  Business Entity Type 1
307510  Business Entity Type 3
```

```
[307511 rows x 32 columns]
```

[25]:
```python
# Save the TARGET column from train_data before encoding
target = train_data['TARGET']

# Identify categorical columns
categorical_cols = train_data.select_dtypes(include=['object']).columns

# Perform one-hot encoding on both train_data and test_data
train_data_encoded = pd.get_dummies(train_data.drop(columns=['TARGET']),␣
↪columns=categorical_cols)
test_data_encoded = pd.get_dummies(test_data, columns=categorical_cols)

# Align train_data and test_data (ensure they have the same columns)
train_data_aligned, test_data_aligned = train_data.align(test_data,␣
↪join='inner', axis=1)

# Add back the TARGET column to the aligned training data
train_data_aligned['TARGET'] = target

# proceed with your further processing (splitting, model building)
print(train_data_aligned.head())
print(test_data_aligned.head())
```

```
   SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
0      100002         Cash loans           M            N               Y
1      100003         Cash loans           F            N               N
2      100004    Revolving loans           M            Y               Y
3      100006         Cash loans           F            N               Y
4      100007         Cash loans           M            N               Y

   CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  AMT_GOODS_PRICE  \
0             0          202500.0    406597.5      24700.5         351000.0
1             0          270000.0   1293502.5      35698.5        1129500.0
2             0           67500.0    135000.0       6750.0         135000.0
3             0          135000.0    312682.5      29686.5         297000.0
4             0          121500.0    513000.0      21865.5         513000.0

   … POS_LOANS_COUNT POS_DPD_MEAN POS_DPD_TOTAL CREDIT_BALANCE_MEAN  \
0  …            19.0          0.0           0.0        71459.926952
1  …            28.0          0.0           0.0        71459.926952
2  …             4.0          0.0           0.0        71459.926952
3  …            21.0          0.0           0.0            0.000000
4  …            66.0          0.0           0.0        71459.926952

  CREDIT_BALANCE_MAX  CREDIT_CARD_MONTHS  PAYMENT_TOTAL  PAYMENT_MEAN  \
0      144501.306629           37.143605     219625.695  11559.247105
```

```
1      144501.306629          37.143605    1618864.650  64754.586000
2      144501.306629          37.143605      21288.465   7096.155000
3           0.000000           6.000000    1007153.415  62947.088438
4      144501.306629          37.143605     806127.975  12214.060227

   TOTAL_INSTALLMENTS  TARGET
0                19.0       1
1                25.0       0
2                 3.0       0
3                16.0       0
4                66.0       0

[5 rows x 103 columns]
   SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
0      100001         Cash loans           F            N               Y
1      100005         Cash loans           M            N               Y
2      100013         Cash loans           M            Y               Y
3      100028         Cash loans           F            N               Y
4      100038         Cash loans           M            Y               N

   CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  AMT_GOODS_PRICE  \
0             0          135000.0    568800.0      20560.5         450000.0
1             0           99000.0    222768.0      17370.0         180000.0
2             0          202500.0    663264.0      69777.0         630000.0
3             2          315000.0   1575000.0      49018.5        1575000.0
4             1          180000.0    625500.0      32067.0         625500.0

   … CREDIT_DURATION_MEAN POS_LOANS_COUNT POS_DPD_MEAN POS_DPD_TOTAL  \
0  …          -735.000000             9.0     0.777778           7.0
1  …          -190.666667            11.0     0.000000           0.0
2  …         -1737.500000            36.0     0.944444          34.0
3  …         -1401.750000            31.0     0.000000           0.0
4  …         -1088.502807            13.0     0.000000           0.0

  CREDIT_BALANCE_MEAN  CREDIT_BALANCE_MAX  CREDIT_CARD_MONTHS  PAYMENT_TOTAL  \
0        62214.550683       130799.447489           36.770972      41195.925
1        62214.550683       130799.447489           36.770972      56161.845
2        18159.919219       161420.220000           96.000000    1509736.545
3         8085.058163        37335.915000           49.000000     492310.665
4        62214.550683       130799.447489           36.770972     133204.050

   PAYMENT_MEAN  TOTAL_INSTALLMENTS
0   5885.132143                 7.0
1   6240.205000                 9.0
2   9740.235774               155.0
3   4356.731549               113.0
4  11100.337500                12.0
```

```
[5 rows x 102 columns]

/tmp/ipykernel_126261/4082780009.py:15: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  train_data_aligned['TARGET'] = target
```

[26]:
```
from sklearn.model_selection import train_test_split

# Train-test split
X = train_data_aligned.drop('TARGET', axis=1)
y = train_data_aligned['TARGET']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42, stratify=y)
```

[27]:
```
# Check for missing values in the training data
print(X_train.isnull().sum())
```

```
SK_ID_CURR           0
NAME_CONTRACT_TYPE   0
CODE_GENDER          0
FLAG_OWN_CAR         0
FLAG_OWN_REALTY      0
                    ..
CREDIT_BALANCE_MAX   0
CREDIT_CARD_MONTHS   0
PAYMENT_TOTAL        0
PAYMENT_MEAN         0
TOTAL_INSTALLMENTS   0
Length: 102, dtype: int64
```

[28]:
```
# Check the data types of the columns in X_train
print(X_train.dtypes)

# Check for non-numeric data
non_numeric_cols = X_train.select_dtypes(exclude=['object']).columns
print("Non-numeric columns:", non_numeric_cols)
```

```
SK_ID_CURR              int64
NAME_CONTRACT_TYPE     object
CODE_GENDER            object
FLAG_OWN_CAR           object
FLAG_OWN_REALTY        object
                         …
CREDIT_BALANCE_MAX    float64
CREDIT_CARD_MONTHS    float64
PAYMENT_TOTAL         float64
```

```
PAYMENT_MEAN          float64
TOTAL_INSTALLMENTS    float64
Length: 102, dtype: object
Non-numeric columns: Index(['SK_ID_CURR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
'AMT_CREDIT',
       'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'REGION_POPULATION_RELATIVE',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH',
       'OWN_CAR_AGE', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
       'FLAG_PHONE', 'FLAG_EMAIL', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'EXT_SOURCE_1',
       'EXT_SOURCE_2', 'EXT_SOURCE_3', 'BASEMENTAREA_AVG',
       'YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BUILD_AVG', 'COMMONAREA_AVG',
       'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG', 'FLOORSMIN_AVG',
       'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG',
       'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG', 'BASEMENTAREA_MODE',
       'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE',
       'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE',
       'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE',
       'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'BASEMENTAREA_MEDI',
       'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI', 'COMMONAREA_MEDI',
       'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI', 'FLOORSMIN_MEDI',
       'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_MEDI',
       'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI', 'TOTALAREA_MODE',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR', 'CREDIT_SUM_TOTAL', 'CREDIT_SUM_MEAN',
       'CREDIT_OVERDUE_MAX', 'CREDIT_DURATION_MEAN', 'POS_LOANS_COUNT',
       'POS_DPD_MEAN', 'POS_DPD_TOTAL', 'CREDIT_BALANCE_MEAN',
       'CREDIT_BALANCE_MAX', 'CREDIT_CARD_MONTHS', 'PAYMENT_TOTAL',
       'PAYMENT_MEAN', 'TOTAL_INSTALLMENTS'],
      dtype='object')
```

[ ]:

[29]:

```
y_train.value_counts()
```

[29]:

```
TARGET
0    226148
1     19860
Name: count, dtype: int64
```

[30]:

```
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

```python
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
from sklearn.ensemble import RandomForestClassifier

def remove_highly_correlated_features(data, threshold=0.7):
    # Select only numerical columns for correlation
    numerical_data = data.select_dtypes(include=[np.number])

    corr_matrix = numerical_data.corr().abs()  # Compute absolute correlation␣
↪matrix
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).
↪astype(bool))  # Upper triangle

    to_drop = []

    while True:
        # Find the column with the highest correlation count
        max_corr_feature = None
        max_corr_count = 0

        for column in upper.columns:
            corr_count = (upper[column] > threshold).sum()
            if corr_count > max_corr_count:
                max_corr_count = corr_count
                max_corr_feature = column

        # If no column has correlations above the threshold, break the loop
        if max_corr_count == 0:
            break

        to_drop.append(max_corr_feature)  # Add the column to the drop list
        upper = upper.drop(columns=max_corr_feature).
↪drop(index=max_corr_feature)  # Drop it from the matrix

    return data.drop(columns=to_drop)

# Remove highly correlated features from the training data
X_filtered = remove_highly_correlated_features(X_train, threshold=0.7)
```

[31]:
```python
X_filtered.head()
```

[31]:
```
        SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
181648      310536         Cash loans           F            N
229245      365516         Cash loans           M            Y
```

```
122525      242055        Cash loans           M            N
306311      454894        Cash loans           M            N
300658      448321        Cash loans           F            N

       FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  \
181648               N             2           90000.0    227520.0
229245               Y             0           90000.0    161730.0
122525               Y             0          135000.0    728847.0
306311               N             0          135000.0    474183.0
300658               Y             0          180000.0    254700.0

        NAME_TYPE_SUITE      NAME_INCOME_TYPE  … CREDIT_SUM_MEAN  \
181648    Unaccompanied  Commercial associate  …   113580.000000
229245    Unaccompanied  Commercial associate  …   378080.200789
122525  Spouse, partner               Working  …   274812.750000
306311    Unaccompanied  Commercial associate  …   399285.000000
300658    Unaccompanied  Commercial associate  …   378080.200789

       CREDIT_OVERDUE_MAX CREDIT_DURATION_MEAN  POS_LOANS_COUNT  POS_DPD_MEAN  \
181648           0.000000          -1175.800000             34.0           0.0
229245           4.772759          -1083.047110             31.0           0.0
122525           0.000000          -1358.500000              5.0           0.0
306311           0.000000          -2005.333333             84.0           0.0
300658           4.772759          -1083.047110             12.0           0.0

        CREDIT_BALANCE_MEAN  CREDIT_CARD_MONTHS  PAYMENT_TOTAL  PAYMENT_MEAN  \
181648        213473.232857           21.000000     550660.815   9660.716053
229245         71459.926952           37.143605     263064.645   9743.135000
122525         71459.926952           37.143605      51958.485  12989.621250
306311         71459.926952           37.143605    1471327.965  17943.023963
300658         71459.926952           37.143605      37066.230   3369.657273

        TOTAL_INSTALLMENTS
181648                57.0
229245                27.0
122525                 4.0
306311                82.0
300658                11.0

[5 rows x 63 columns]
```

[32]:
```
X_train.columns
```

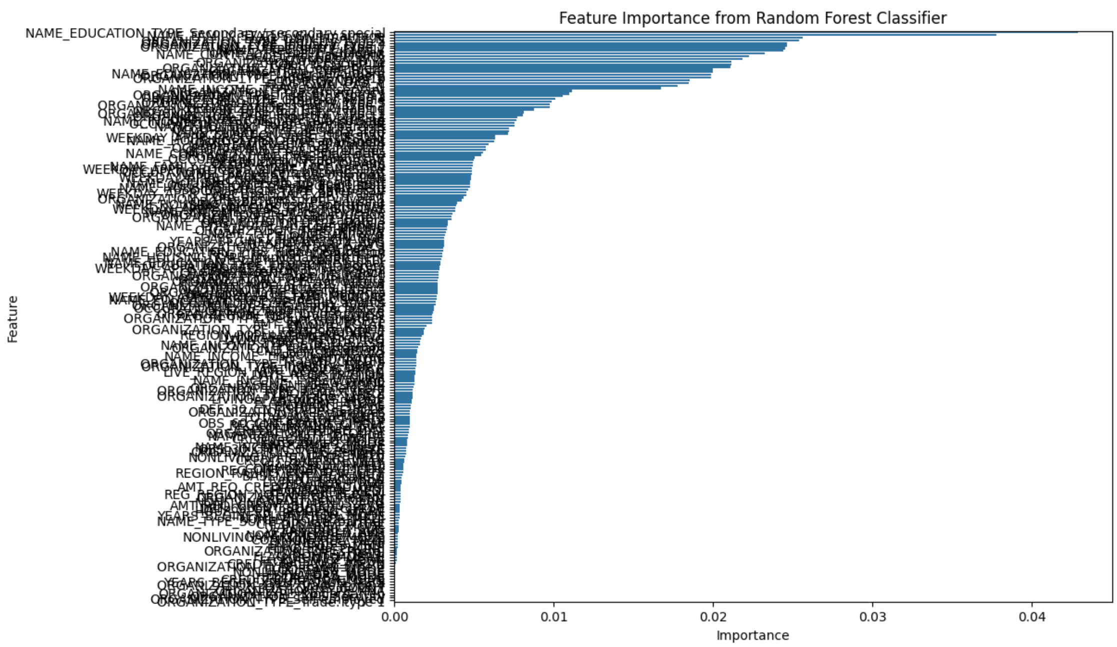
[32]:
```
Index(['SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR',
       'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT',
       'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       …
```

```
        'CREDIT_DURATION_MEAN', 'POS_LOANS_COUNT', 'POS_DPD_MEAN',
        'POS_DPD_TOTAL', 'CREDIT_BALANCE_MEAN', 'CREDIT_BALANCE_MAX',
        'CREDIT_CARD_MONTHS', 'PAYMENT_TOTAL', 'PAYMENT_MEAN',
        'TOTAL_INSTALLMENTS'],
       dtype='object', length=102)
```

[36]:

```python
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.exceptions import NotFittedError

# Preprocessing
numerical_cols = X_train.select_dtypes(include=['number']).columns
categorical_cols = X_train.select_dtypes(include=['object']).columns

numerical_imputer = SimpleImputer(strategy='mean')
categorical_imputer = SimpleImputer(strategy='most_frequent')

preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('imputer', numerical_imputer),
            ('scaler', StandardScaler())
        ]), numerical_cols),
        ('cat', Pipeline(steps=[
            ('imputer', categorical_imputer),
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ]), categorical_cols)
    ]
)

# Build the pipeline with Random Forest Classifier
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])

# Fit the pipeline
pipeline.fit(X_train, y_train)

# Train model
```

```python
# model = RandomForestClassifier(random_state=42)
# model.fit(X_train, y_train)


# Feature Importance Extraction
try:
    onehot_encoder = pipeline.named_steps['preprocessor'].named_transformers_['cat'].named_steps['onehot']
    categorical_feature_names = onehot_encoder.get_feature_names_out(categorical_cols)

    feature_names = list(categorical_feature_names) + list(numerical_cols)
    importance = pipeline.named_steps['classifier'].feature_importances_

    feature_importance_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance': importance
    }).sort_values(by='Importance', ascending=False)

    # Visualization
    plt.figure(figsize=(10, 8))
    sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
    plt.title('Feature Importance from Random Forest Classifier')
    plt.show()

except NotFittedError:
    print("The OneHotEncoder instance is not fitted. Ensure that the pipeline is fitted correctly.")
```



[37]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score


# Make predictions on the test set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}, Precision: {precision:.2f}, Recall: {recall:.
↪2f}, F1-Score: {f1:.2f}")
```

```
Accuracy: 0.92, Precision: 0.67, Recall: 0.00, F1-Score: 0.00
```

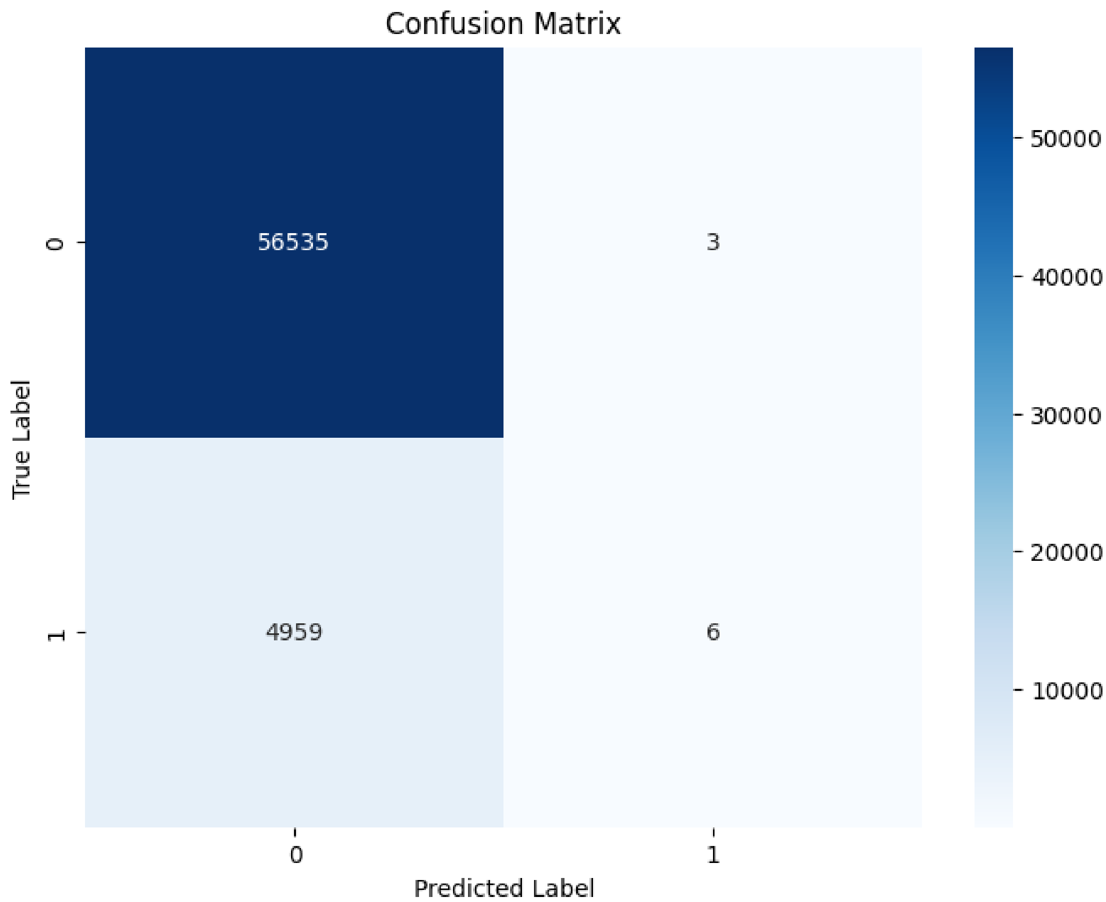
[38]:
```python
# Performance evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.92
Classification Report:
               precision    recall  f1-score   support

           0       0.92      1.00      0.96     56538
           1       0.67      0.00      0.00      4965

    accuracy                           0.92     61503
   macro avg       0.79      0.50      0.48     61503
weighted avg       0.90      0.92      0.88     61503
```

[39]:
```python
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

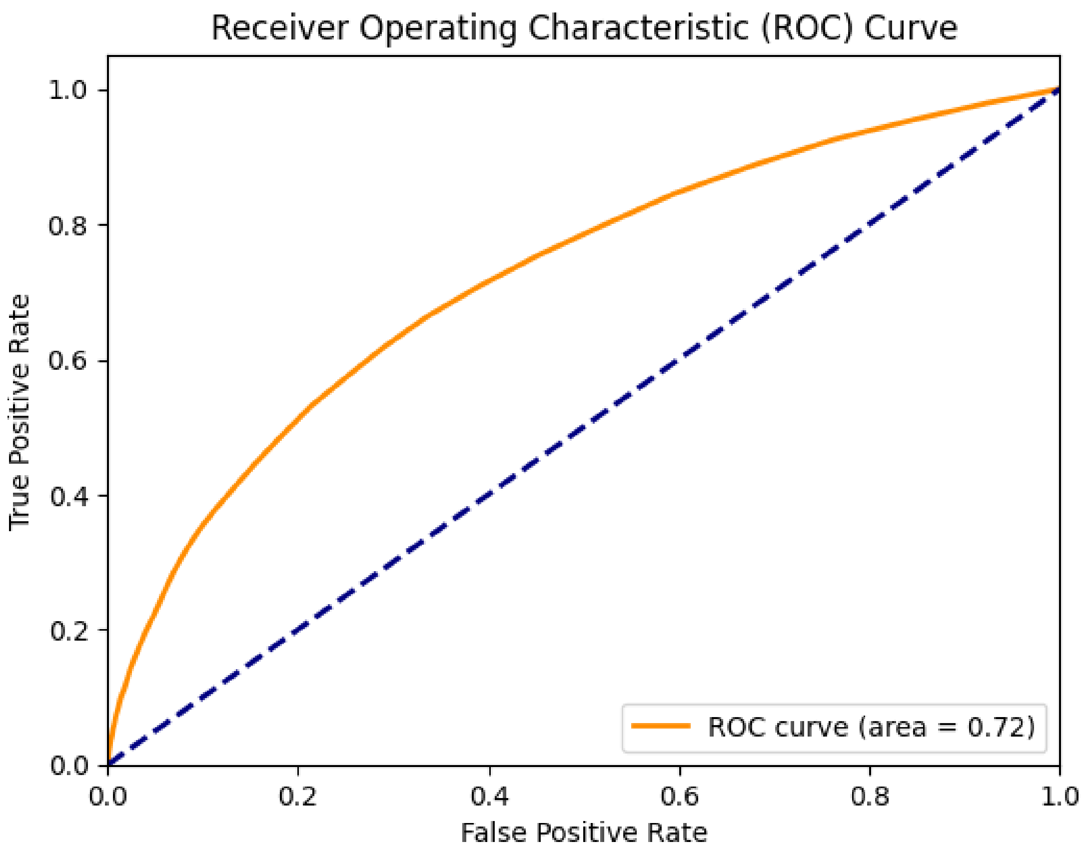
```
Confusion Matrix:
[[56535     3]
 [ 4959     6]]
```

[40]:
```
# Plot confusion matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Confusion matrix visualization
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
plt.show()
```



[41]:
```
# ROC curve and AUC
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, pipeline.predict_proba(X_test)[:, 1])
roc_auc = auc(fpr, tpr)

# Plot ROC Curve
```

```
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:
 ↪.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



[ ]:
```
#Hyperparameter Tuning

from sklearn.model_selection import GridSearchCV

# Define the parameter grid for Random Forest
param_grid = {
    'classifier__n_estimators': [50, 100, 200],
```

```
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 5, 10],
}

# Perform Grid Search
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print("Best Hyperparameters:", grid_search.best_params_)
```

[ ]:
```
## Model Comparison and Selection

from sklearn.linear_model import LogisticRegression

# Define other models
models = {
    'Random Forest': RandomForestClassifier(random_state=42),
    'Logistic Regression': LogisticRegression()
}

# Iterate through models and evaluate performance
for model_name, model in models.items():
    pipeline = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', model)
    ])

    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"{model_name} Accuracy: {accuracy:.2f}")
```

[22]:
```
import pandas as pd
sales_exp_pd = pd.read_csv('/home/ignatiusvmk/sales_export')
accounts_pd = pd.read_csv('/home/ignatiusvmk/accounts_export')
```

[ ]:
```
# sales_exp_pd.head(15)
accounts_pd.head(20)
```