Home Loan Default Risk Prediction

(Team - Caesar)

Team Member 1 - Dhruv Awasthi (Point of Contact)
Team Member 2 - Ashok Senapati

Key Features

- Command line interface
- Built as a pipeline
- Modular approach
- Abstract components
- Easily scalable (current size ~24.4kB)
- Log everything
- Exception handling
- Save model for later use
- Single file configuration
- Pickle dump important objects
- Version control system
- No hard coding, can be used for n-number of features and any dataset

Reduce space complexity

Reduced the dataset size to reduce the space complexity and speed up the training

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184506 entries, 0 to 184505
Columns: 122 entries, SK_ID_CURR to TARGET
dtypes: float64(65), int64(41), object(16)
memory usage: 171.7+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184506 entries, 0 to 184505
Columns: 122 entries, SK_ID_CURR to TARGET
dtypes: float32(50), float64(14), int16(3), int32(1), int64(1), object(16), uint32(1), uint8(36)
memory usage: 87.6+ MB
```

Preprocessing Pipeline

Proceeded column-wise instead of process-wise because each column encodes different information and it is important to analyse each column separately.

CNT_CHILDREN

In [5]:
```
column_name = "CNT_CHILDREN"
index_of_outliers, less_than_mean, more_than_mean, unique_values, iqr_range, lower_bound, upper_bound, column_mean_
```

```
CNT_CHILDREN
Min value: 0
Max value: 19
Std: 0.7196143256712833
Min std: 0.0
Max std:  402198.1853209468

Pearson's correlation: 0.020010553326490807
Spearman's correlation: 0.020319501696053968
Missing value(s): 0


Lower bound: -1.7
Upper bound: 2.7
Number of outlier(s): 2502
Mean with outliers: 0.4164634212437536
Mean without outliers: 0.37834333311355794
Less than mean: 0
More than mean: 11

Unique values: 14
```

Variant I

- For categorical columns, replace less-occuring values with another category.

```
In [7]: df[column_name].value_counts()

Out[7]: 0     129082
        1      36984
        2      15938
        3       2190
        4        237
        5         48
        6         12
        7          5
        14         2
        12         2
        8          2
        9          2
        10         1
        19         1
        Name: CNT_CHILDREN, dtype: int64

In [8]: df[column_name].replace([3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], 3, inplace=True)
        df[column_name].value_counts()

Out[8]: 0    129082
        1     36984
        2     15938
        3      2502
        Name: CNT_CHILDREN, dtype: int64
```

Variant II

- For numerical columns, trim the extreme scores before dealing with them

In [14]:
```
lowest_outlier_value = more_than_mean[0]
index_lowest_outlier_value = np.where(unique_values == lowest_outlier_value)[0][0]
print(f"Lowest outlier value in unique values: {lowest_outlier_value}")
print(f"Index of lowest outlier value in unique values: {index_lowest_outlier_value}")
print(f"Outliers will be trimmed to: {unique_values[index_lowest_outlier_value - 1]}")
```

```
Lowest outlier value in unique values: 356625.0
Index of lowest outlier value in unique values: 1548
Outliers will be trimmed to: 355500.0
```

Variant III

- For numerical columns, remove the extreme scores that are too extreme

```
In [26]: print(f"Before deleting: {len(index_of_outliers)}")
         index_of_outliers = index_of_outliers.drop(to_delete_indices)
         print(f"After deleting: {len(index_of_outliers)}")

         Before deleting: 3191
         After deleting: 3190
```

Variant IV

- Drop columns with extreme low standard deviation

```
In [85]: df[column_name].value_counts()

Out[85]: 1    184470
         0         1
         Name: FLAG_MOBIL, dtype: int64
```

Variant V

- Drop columns with bad correlation

FLAG_DOCUMENT_4

In [108]:
```
column_name = "FLAG_DOCUMENT_4"
index_of_outliers, less_than_mean, more_than_mean, unique_values, iqr_range, lower_bound, upper_bound, column_mean_
```

```
FLAG_DOCUMENT_4
Min value: 0
Max value: 1
Std: 0.009599344427845154
Min std: 0.0
Max std:  402198.1853209468

Pearson's correlation: -0.002844925517529631
Spearman's correlation: -0.0028449255175296466
Missing value(s): 0


Lower bound: 0.0
Upper bound: 0.0
Number of outlier(s): 17
Mean with outliers: 9.215540654086551e-05
Mean without outliers: 0.0
Less than mean: 0
More than mean: 1

Unique values: 2
```

Variant VI

- Feature engineering with categories

```
Out[343]: Business          50788
          XNA               33230
          Self-employed     22877
          Education         10246
          Other              9958
          Trade              8667
          Industry           8531
          Medicine           6749
          Security           6480
          Government         6224
          Transport          5406
          Realtor            4311
          Services           2821
          Housing            1929
          Dining             1674
          Bank               1493
          Agriculture        1439
          Electricity         560
          Mobile              533
          Culture             291
          Advertising         264
          Name: ORGANIZATION_TYPE, dtype: int64
```

Variant VII

- Encoding the categorical columns using,
  - Label Encoding
  - One Hot Encoding

In [330]:
```
mapping_dict = {
        "Working": 2,
        "Commercial associate": 2,
        "Pensioner": 1,
        "State servant": 3,
        "Unemployed": 0,
        "Businessman": 3
    }

df[column_name].replace(mapping_dict, inplace=True)
df[column_name].value_counts()
```

Out[330]:
```
2    138242
1     33223
3     12985
0        21
Name: NAME_INCOME_TYPE, dtype: int64
```

Variant VIII

- Drop columns with large number of missing values

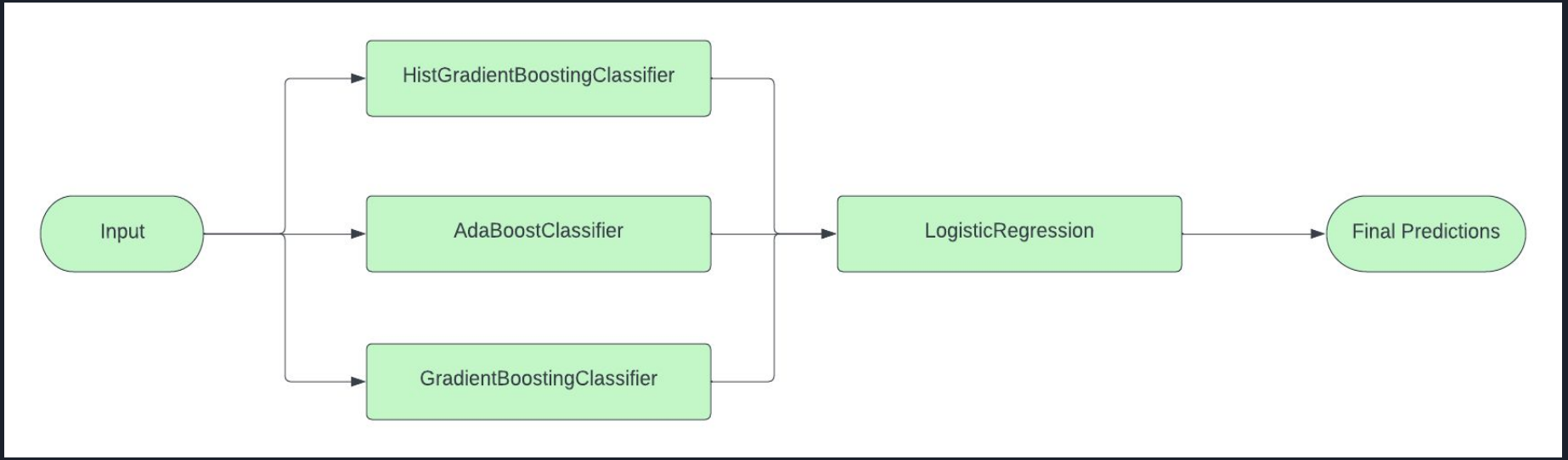
COMMONAREA_AVG

In [208]:
```
column_name = "COMMONAREA_AVG"
index_of_outliers, less_than_mean, more_than_mean, unique_values, iqr_range, lower_bound, upper_bound, column_mean_
```

```
COMMONAREA_AVG
Min value: 0.0
Max value: 1.0
Std: 0.075887531042099
Min std: 0.0
Max std:  402198.1853209468

Pearson's correlation: -0.021531113415197287
Spearman's correlation: -0.016316561765030935
Missing value(s): 128945


Lower bound: -0.06615000125020742
Upper bound: 0.12525000143796206
Number of outlier(s): 4226
Mean with outliers: 0.04450373724102974
Mean without outliers: 0.028348391875624657
Less than mean: 0
More than mean: 1576

Unique values: 2829
```

Model - Stacked Architecture

- It is a method for combining the estimators to reduce the biases.
- The predictions of each individual estimator are stacked together and used as input to a final estimator to compute the prediction.
- The final estimator is trained through cross-validation.



Thank you!