Bank Loan Case Study

A DATA ANALYTICS PROJECT

Project By:

Yaadhav R

Project Description

In this project, I'll conduct Exploratory Data Analysis (EDA) for a finance company specializing in lending to urban customers. By analyzing loan application data, I aim to identify patterns influencing loan default likelihood. With a focus on customers with payment difficulties and those with timely payments, I'll investigate how various customer and loan attributes impact approval outcomes. Through this analysis, I aim to mitigate the risk of rejecting capable applicants while minimizing financial losses from defaults. The project aims to enhance decision-making processes, ensuring a balanced approach to loan approvals and reducing the company's exposure to risk.

Tech-Stack Used

Jupyter Notebook - Version 7.0.7 is used to implement Python with various other libraries :

- Pandas - Data Manipulation and Analysis.
- Numpy - Mathematical Computations.
- Matplotlib - Data Visualization
- Seaborn - Advanced Data Visualization

Jupyter Notebook Github Link : 🔗

Tasks

- A. Identify Missing Data and Deal with it Appropriately
- B. Identify Outliers in the Dataset
- C. Analyze Data Imbalance
- D. Perform Univariate, Segmented Univariate, and Bivariate Analysis
- E. Identify Top Correlations for Different Scenarios

Handling Missing Data

Approach

- First the columns with more than 10,000 missing values are identified and dropped.

```
# Removes the columns with more than 10K missing values

data.dropna( axis=1, thresh=40000, inplace=True)
data.shape
```

- Then the missing values in the remaining columns are filled with the respective column means and the duplicate rows are dropped as well.

```
# Filling the Missing the values with the respective column mean

for col in data.select_dtypes(include=['int64', 'float64']):
    data[col].fillna( value=data[col].mean(), inplace=True)
data.drop_duplicates()
data.info()
```

Insights

- Initially, there were 14,88,212 null values in the given dataset.
- After the first step it significantly reduced to 51,379, which is almost 97% of the nulls were removed.
- At the same time, the number of columns in the data was also reduced from 122 to 72.
- After replacing the missing values with th respective column mean value, there were no duplicate rows in the data.
- This was noticed when the shape of the data frame remained the same (4999, 72) before and after executing the drop_duplicates function.

Outliers Detection

Approach

- First the feature columns are selected based the previous application data. The common columns in data and prev are chosen.

```
# Feature Columns are selected with the help of the previous application data

feature_columns = data.select_dtypes(include=['int64','float64']).columns.intersection(prev.columns)
feature_columns = feature_columns.to_list()
```

- Then the outliers in these feature columns are detected with the help of box plot. So inter-quartile range and z-score methods are employed to remove them.
- In inter-quartile range method, the inter-quartile range(iqr) is calculated from the difference of Q3(75%) and Q1(25%), with this iqr the upper bound and lower bound are calculated.

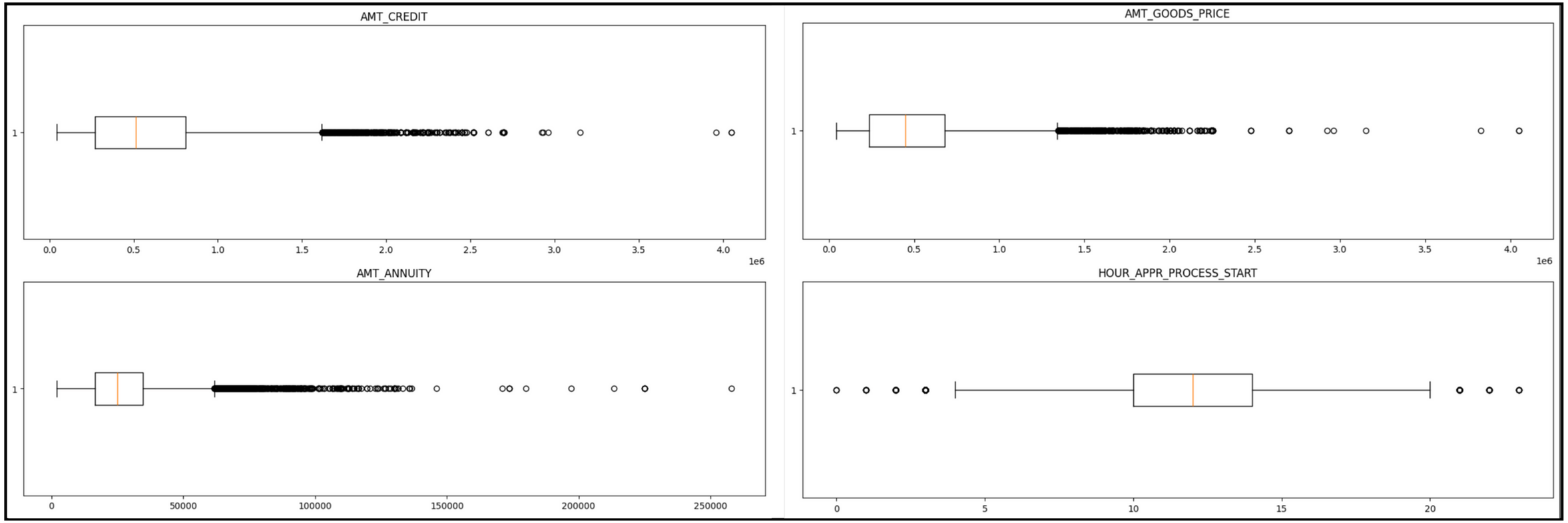
- In each column, the values more than the upper bound and less than the lower bound are considered as outliers and the respective rows are filtered out from the data.

```
for col in feature_columns:
    q1, q3 = np.percentile(data[col], [25, 75])
    iqr = q3 - q1

    lower_bound = q1 - (1.5 * iqr)
    upper_bound = q3 + (1.5 * iqr)

    data = data[(data[col] >= lower_bound) & (data[col] <= upper_bound)]
data.shape
```

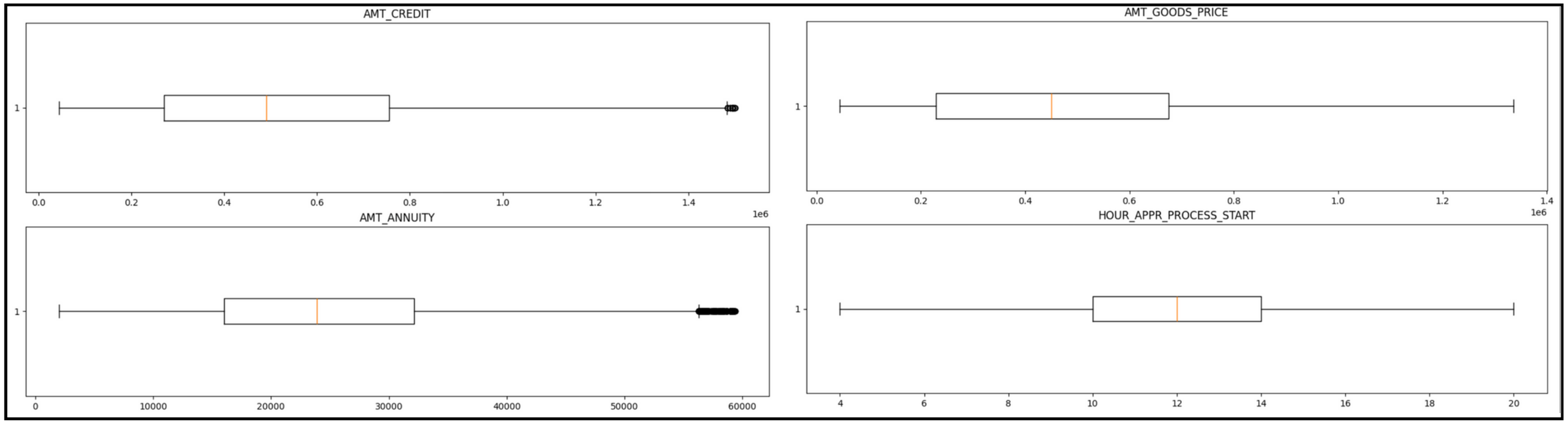
- In Z-Score method, the z-score of each column is calculated and the entire row of the values with score > 3 are removed.

```
for col in feature_columns:
    z_scores = (data[col]-data[col].mean())/data[col].std()
    data.drop( z_scores[abs(z_scores) > 3].index, axis=0, inplace=True)

data.shape
```

With the Presence of Outliers



After the Removal of Outliers



Insights

- The shape of the data frame was (49999, 72) before the removal of outliers.
- 3417 rows were dropped from the data after applying the inter-quartile range method and the new shape was (46582, 72).
- Further 36 rows were dropped from the data after applying z-score method and the final shape was (46546,72).
- Hence a total of 3453 rows were dropped from the data.
- Even the though two methods were employed to remove the outliers, still there were outliers in the columns ‘amt_credit’ and ‘amt_annuity’ but the outliers were significantly less so they were neglected.

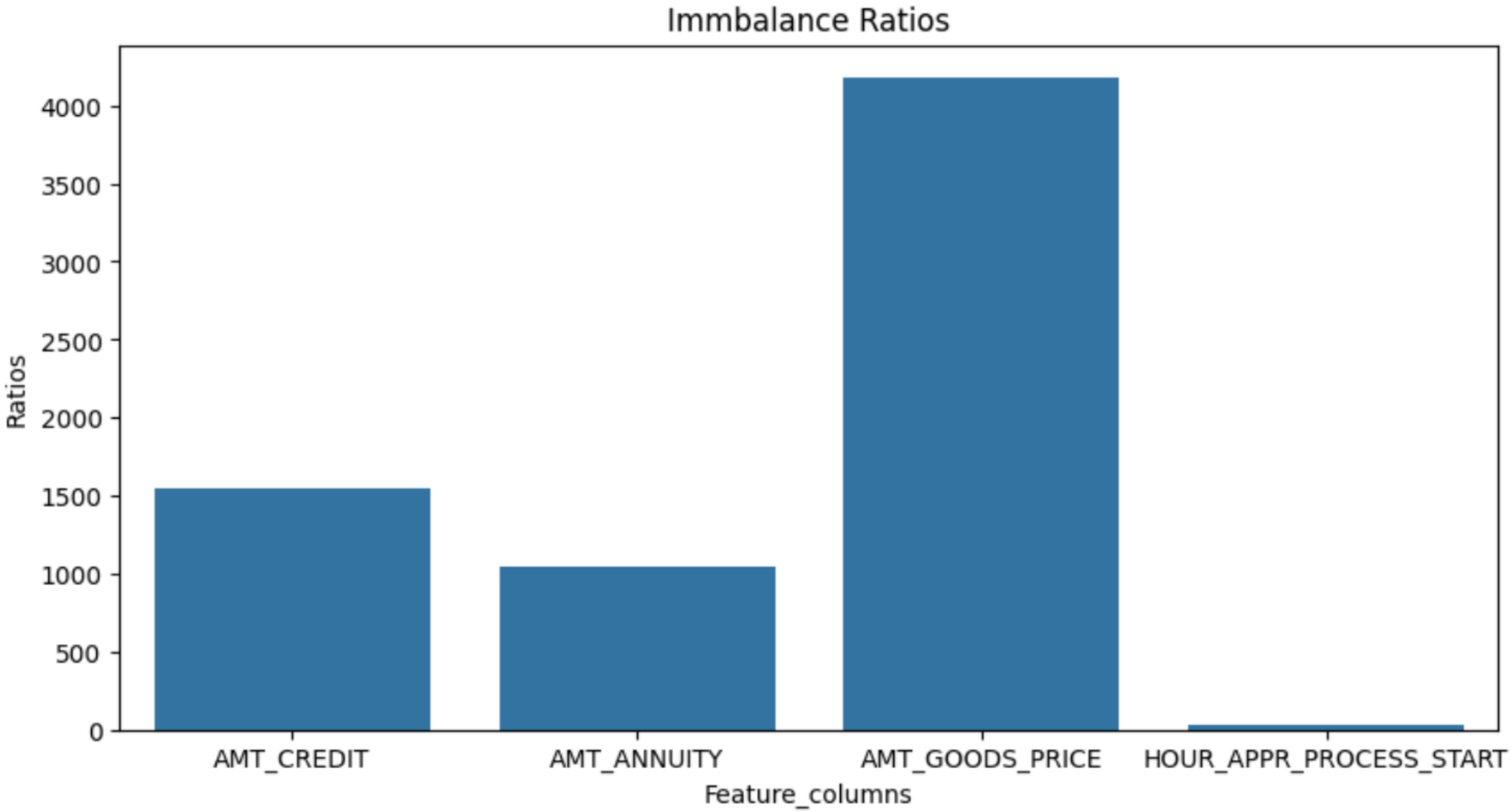
Analysing Data Imbalance

Approach

- First the occurrence frequencies in each column is calculated.
- Then to calculate the imbalance ratio, the maximum of the frequencies is divided by the minimum frequencies.
- This is applied to the four feature columns and their imbalance ratios are found.

```
# Data Imbalance Ratio Calculation

ratios = []
for col in feature_columns:
    freq = data[col].value_counts()
    imbalance_ratio = max(freq)/min(freq)
    ratios.append((col, imbalance_ratio))
ratios
```

Bar Chart of Imbalance Ratios



Insights

- The imbalance ratios of the four feature columns as follows:
  amt_credit = 1551.0
  amt_annuity = 1039.0
  amt_goods_price = 4181.0
  hour_appr_process_start = 34
- The imbalance is highest for amt_goods_price with 4181 and lowest for hour_appr_process_start with just 34.
- Hence the range of the imbalance ratio is 4147, which quite large.
- The overall average imbalance ratio is 1701.25, which is again very high ratio which may result in poor prediction of models.

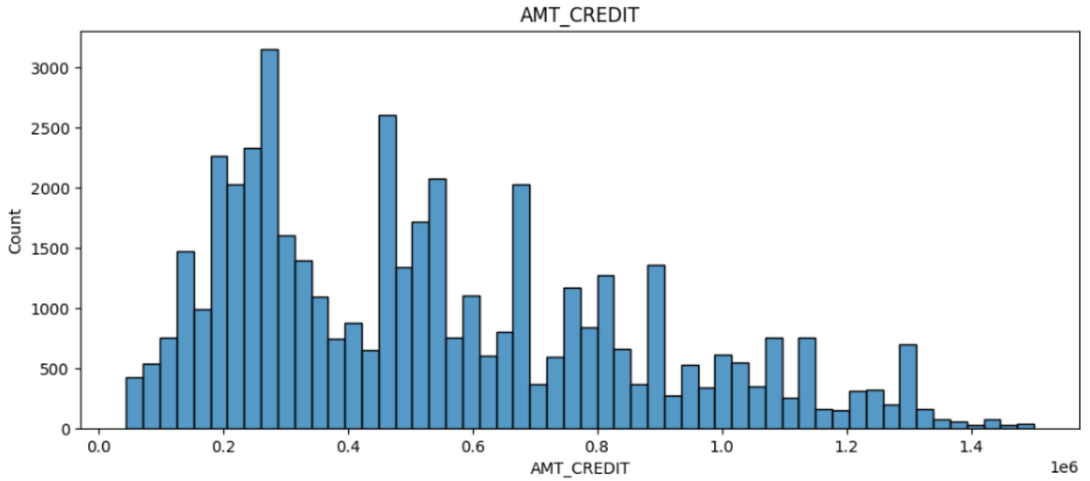
Univariate and Bivariate Analysis

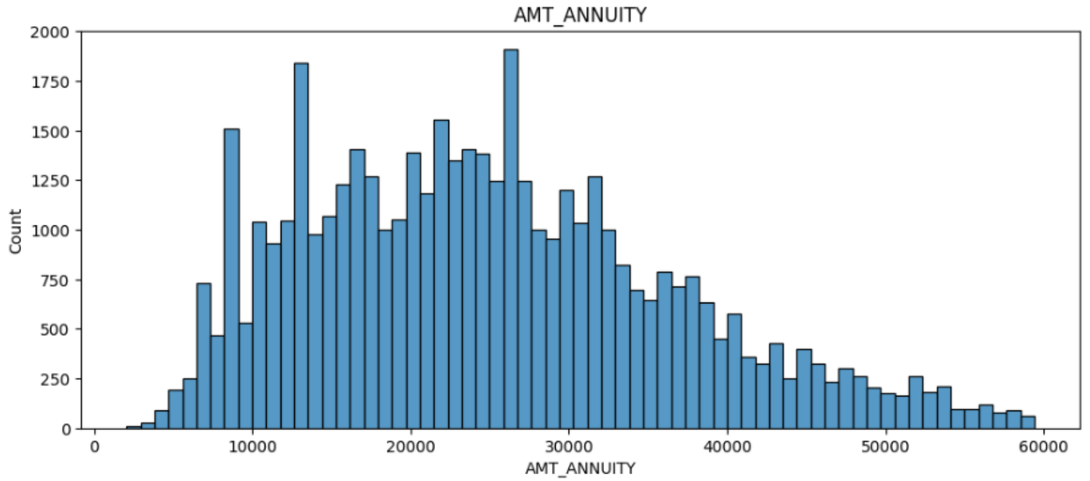
Approach

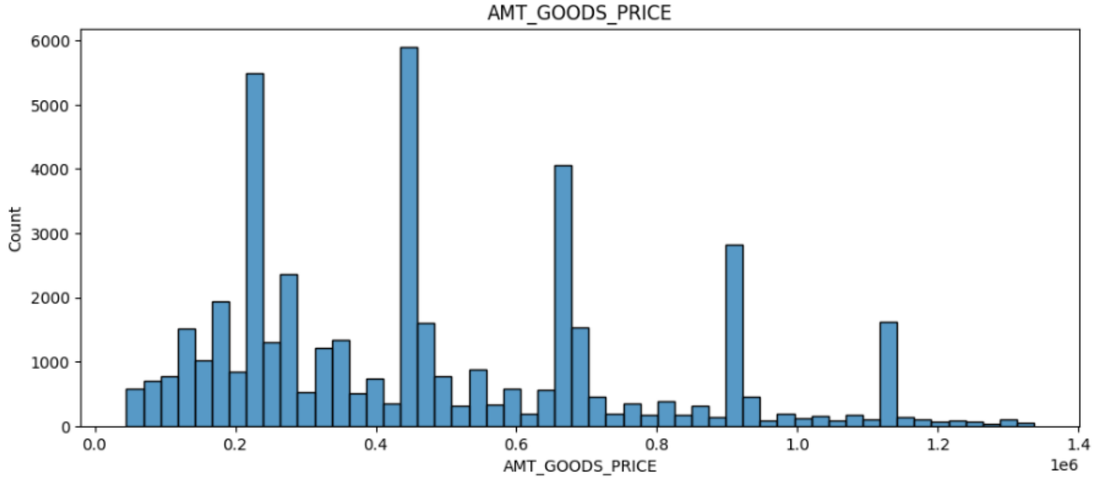
- For every graphs, first a figure size is defined then the type of graph is plotted on this figure with the required parameters and finally the plot is displayed.
- A sample of one graph is shown below.

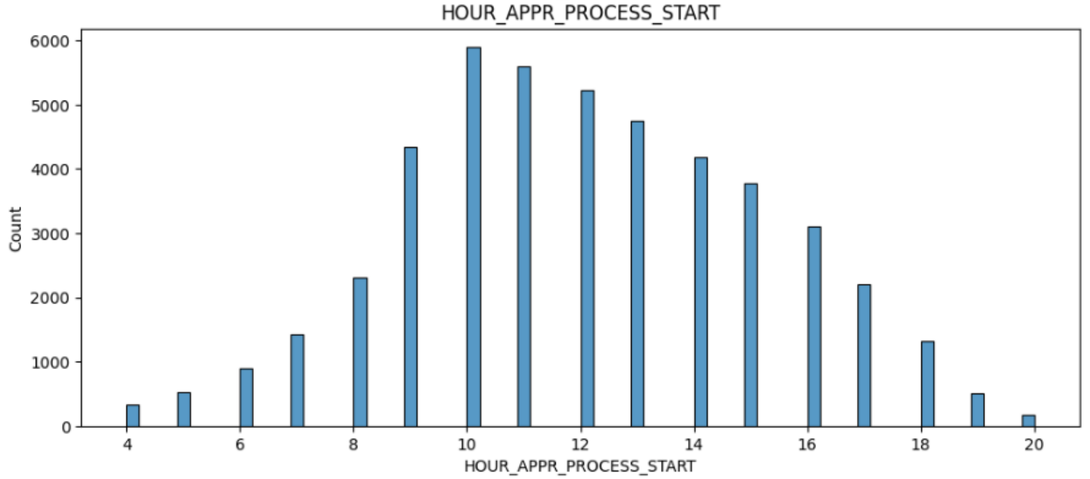
```
plt.figure(figsize=(20,20))

i=1
for col in feature_columns:
    plt.subplot( 2, 2, i)
    sns.histplot(data[col])
    plt.title(col)
    i+=1
plt.show()
```

Univariate Analysis of Feature Columns

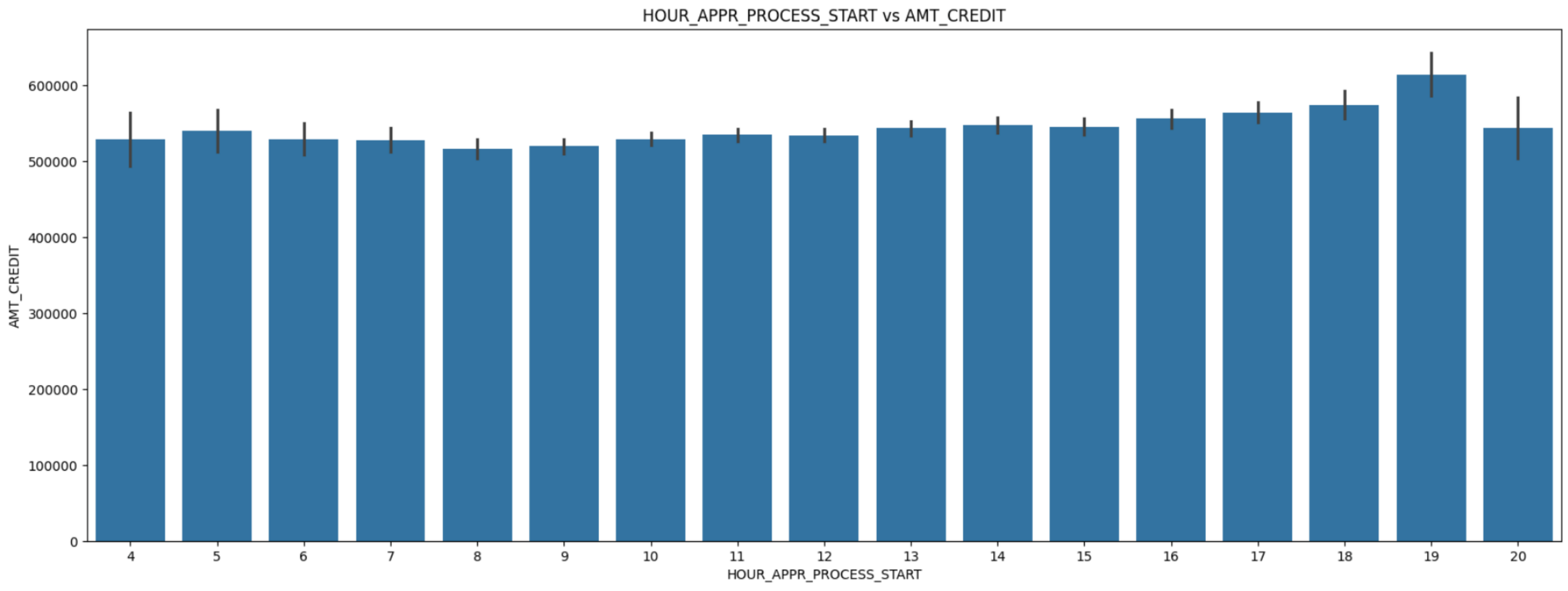




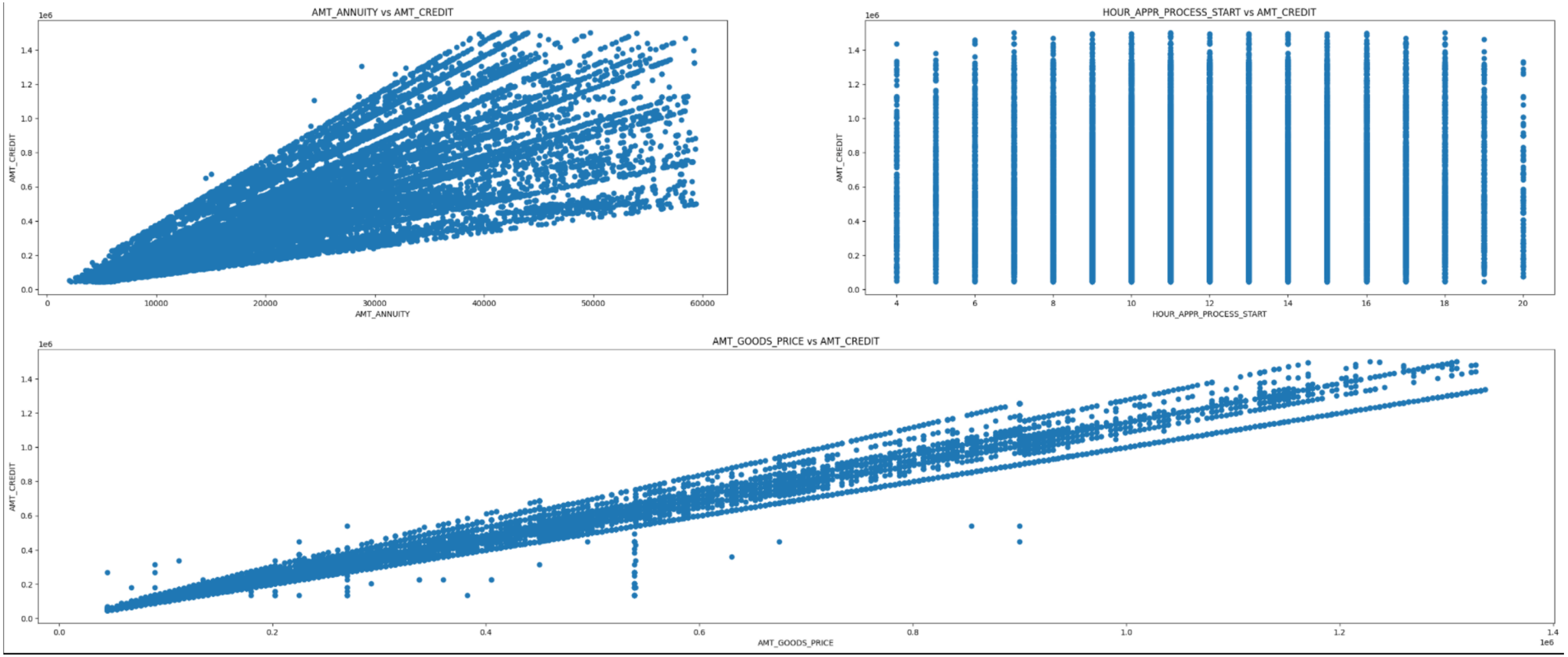




Bivariate Analysis with Bar charts



Bivariate Analysis with Scatter charts



Insights

- From Univariate Analysis of Feature Columns, the graphs of ‘amt_credit’ & ‘amt_annuity’ were similar and that of ‘amt_goods_price’ had sudden hikes.
- For the bivariate analysis, ‘amt_credit’ was chosen as the target variable and it was compared with other features.
- The Bar chart suggests a very similar values for all ‘hour_appr_process_start’, which implies that this feature does have any impact on the target variable.
- Even from the Scatter chart the same was observed.
- But for the other two features, it was observed that the rise in these features also resulted in the rise of target variable.

Correlations

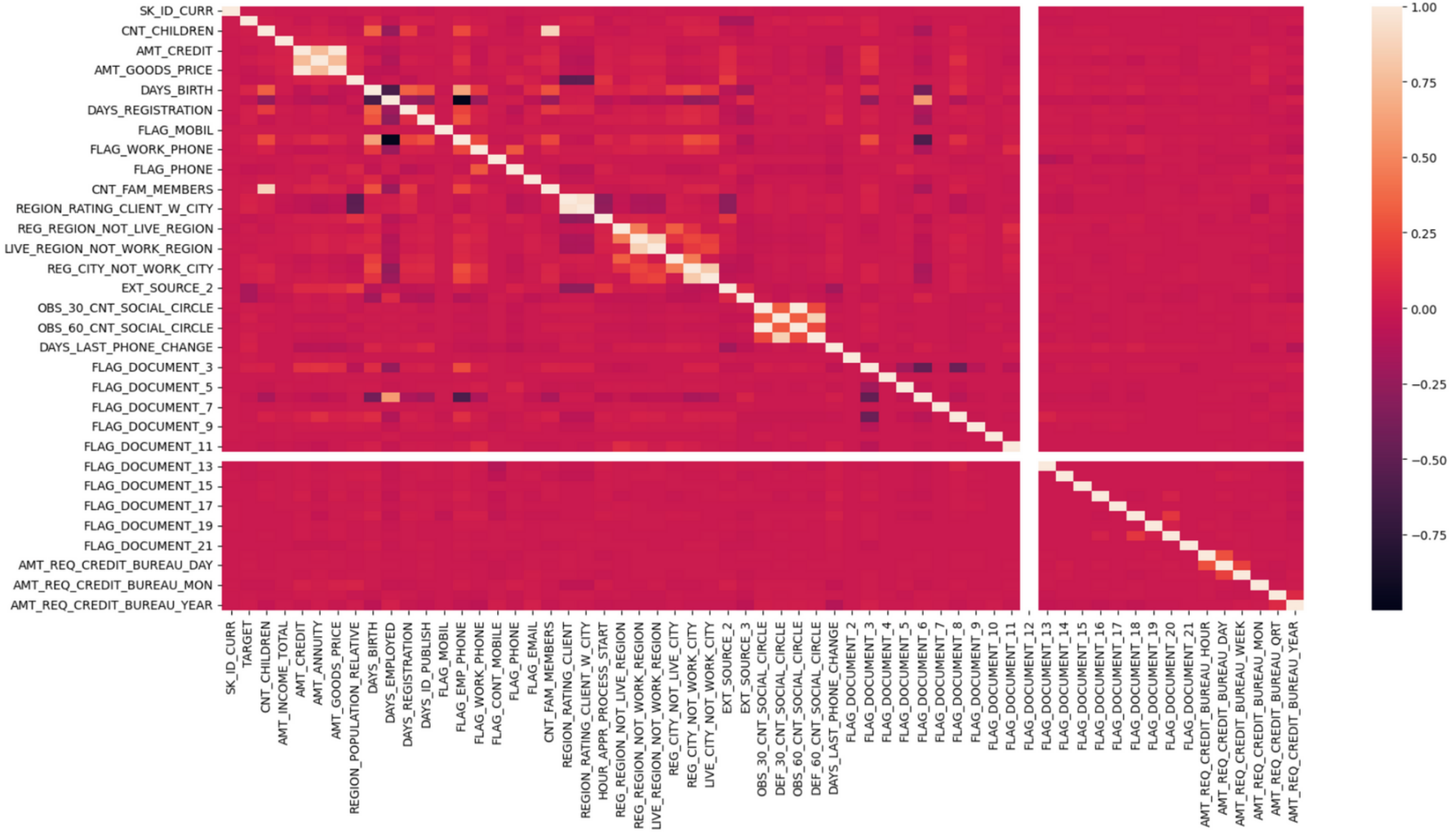
Approach

- The Correlation of a data frame can be found directly by using the corr() function.
- Then heatmap is applied upon this for visualization.
- Here Correlation of the entire data is found first, then it is narrowed down to the feature columns.

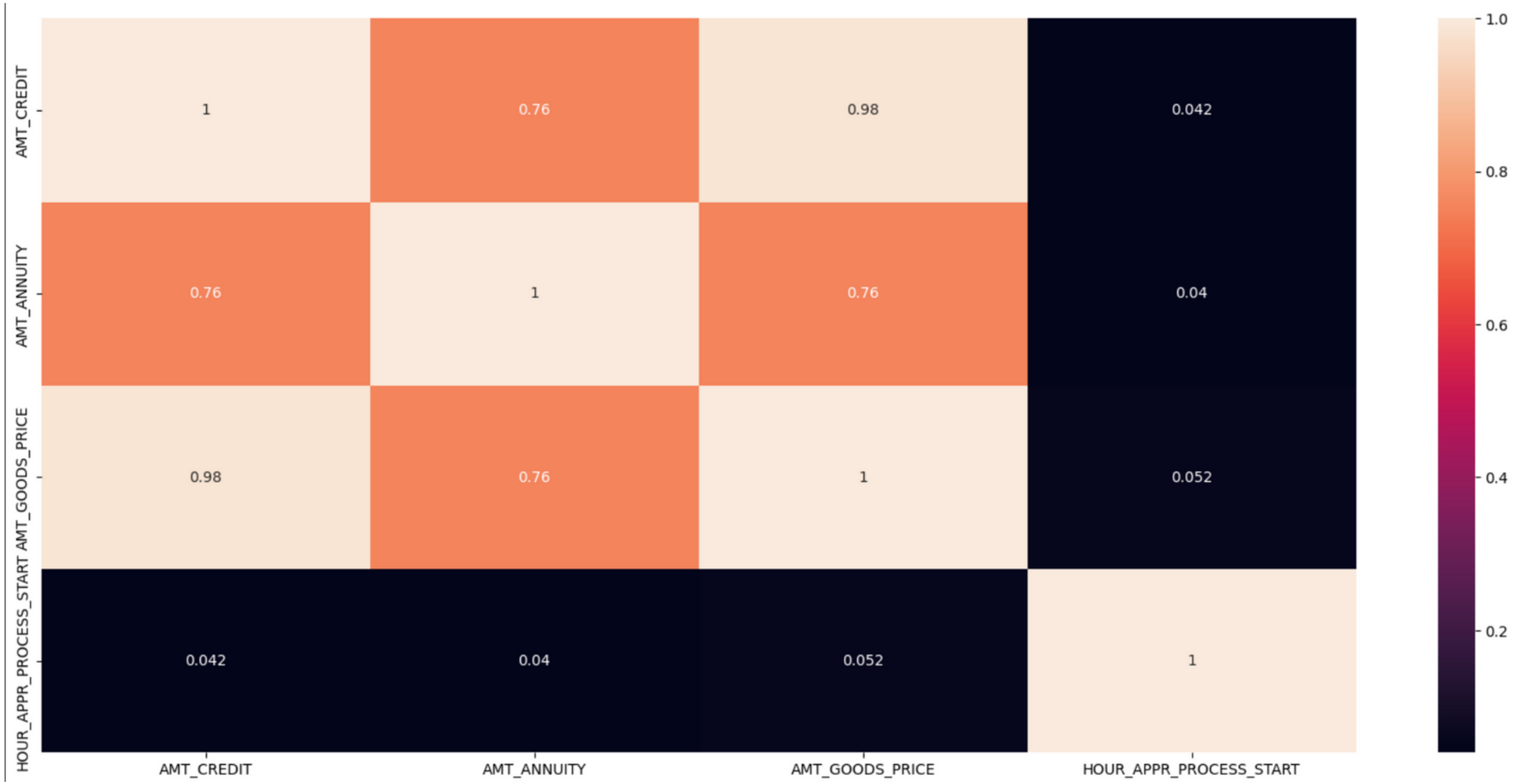
```
plt.figure( figsize=(5,5) )
sns.heatmap( data.select_dtypes(include=['int64','float64']).corr() )
plt.show()
```

```
plt.figure( figsize=(5,5) )
sns.heatmap( data[feature_columns].corr(), annot=True )
plt.show()
```

Correlation of entire data



Correlation of Feature Columns



Insights

- Since there were 72 columns in our data, hardly any observation can be made from the heatmap.
- But overall it was observed that the correlation between most of the columns were close to zero suggesting very low level of dependency between the data.
- So the correlation of feature columns were found.
- From this heat map, it was observed only ‘hour_appr_process_start’ was irrelavant but all the other 3 feature columns had correlation coefficient more than 0.75.
- Finally the correlation coeff of ‘amt_credit’ and ‘amt_goods_price’ was 0.98 which is significantly close to 1. Hence they were highly related which was same as observed from scatter plot.