Data Science Project

Loan Classification

name	Section	number	score
غادة احمد السيد	5	146	
هاجر حمدي محمد	8	233	
نادين أشرف حامد	8	224	
امال عبد العظيم محمد	2	47	
دينا يسرى احمد	3	87	
رضوى سيد سعيد	4	91	
منه الله هاني	8	214	
الاء احمد محمد	2	39	
كريم إيهاب محمد	6	151	

Problem Definition

Predict loan and whose applicant deserve it. Its primary objective of the project could be to analyse and understand the factors affecting loan approval, predict loan defaulters, or optimize loan processes for better efficiency. Customer info like [id, status, education, loan amount, loan_ status credit history] all this from LOAN Dataset.

Still some banks want automatic process to predict person deserves to get loan or not and more accurate and efficient system.

The company seeks to automate (in real time) the loan qualifying procedure based on information given by customers while filling out an online application form. It is expected that the development of ML models that can help the company predict loan approval in accelerating decision.

Main step in project:

1- **Loading Data (data have 34 columns, 148670 entries)**
2- **Feature exploration and visualization using (cufflinks, Plotly , Matplotlib, Seaborn)**
3- **Preprocessing to data frame**

Drop two columns (year ,id):

81]:

```
df=df.drop(['ID','year'],axis=1)
```

Divide data to categorical ,numerical values

```
num=[]
for x in df:
    if df[x].dtype == 'int64' or df[x].dtype == 'float64':
        num.append(x)


print("numerical col :",num)
```

```
numerical col : ['loan_amount', 'rate_of_interest', 'Interest_rate_spread', 'Upfront_charges', 'term', 'property_valu
e', 'income', 'Credit_Score', 'LTV', 'Status', 'dtir1']
```

```
cat=[]
for x in df.columns:
    if df[x].dtype == 'object':
        cat.append(x)


print("categorical col :",cat)
```

```
categorical col : ['loan_limit', 'Gender', 'approv_in_adv', 'loan_type', 'loan_purpose', 'Credit_Worthiness', 'open_cre
dit', 'business_or_commercial', 'Neg_ammortization', 'interest_only', 'lump_sum_payment', 'construction_type', 'occupan
cy_type', 'Secured_by', 'total_units', 'credit_type', 'co-applicant_credit_type', 'age', 'submission_of_application',
'Region', 'Security_Type']
```

Remove null values in

- **categorical using -> mode**

```
for i in cat:
    mode=df[i].mode()
    mode=mode[0]
    df[i].fillna(value=mode,inplace=True)
```

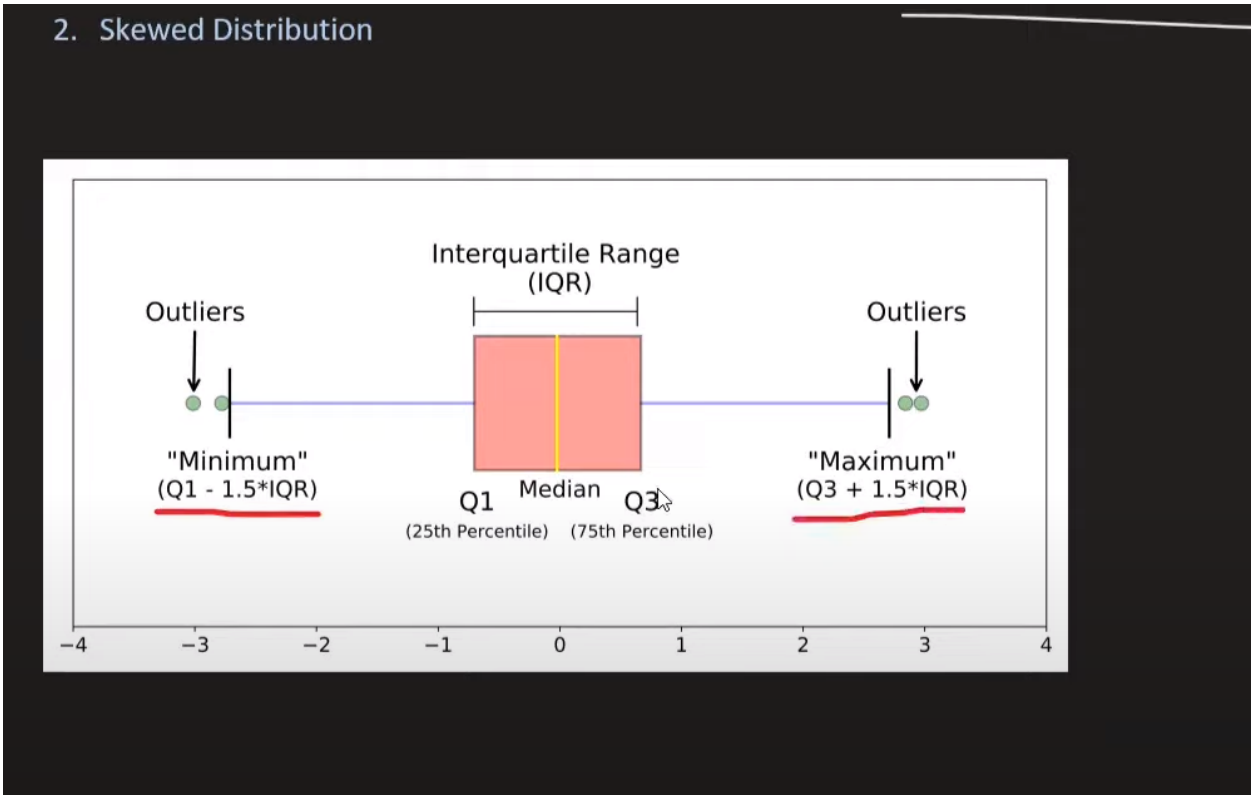
- **Numerical->mean technique**

```
from sklearn.impute import SimpleImputer
mean_imputer = SimpleImputer(strategy='mean')
df[num] = mean_imputer.fit_transform(df[num])
```

```
df.Status.unique()
```

```
array([1., 0.])
```

Remove outliers from numerical values using IQR



IQR = Q3 — Q1

To identify outliers using the IQR method, we establish two boundaries:

- Lower Bound: Q1–1.5 * IQR
- Upper Bound: Q3 + 1.5 * IQR

4- Encoding using binary (label encoder)

```
# adding a new feature
from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()
for i in cat:
    df[i] = label.fit_transform(df[i])
```

5- Split data and standardization using standardisable Range in data from 0 to 1

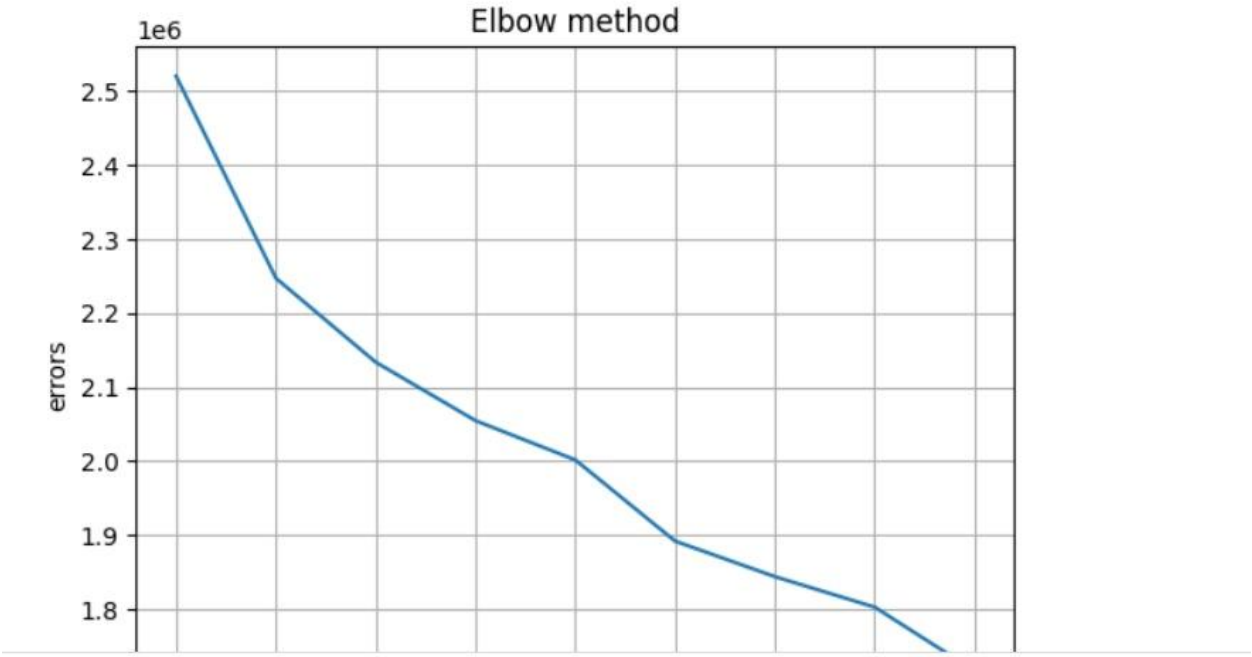
```
# Split the dataset into input (X) and output (y)

X = df.drop(["Status"], axis=1)
y = df["Status"]


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

6-Ml models and accuracy in classification

	Model	Training Score	Testing Score
0	KNN	93%	92%
1	Logistic Regression	75%	76%
2	Random Forest	100%	100%
3	Naive Bayes	74%	75%
4	Decision Tree	100%	100%
5	SVM	76%	76%
6	BernoulliNB	91%	91%

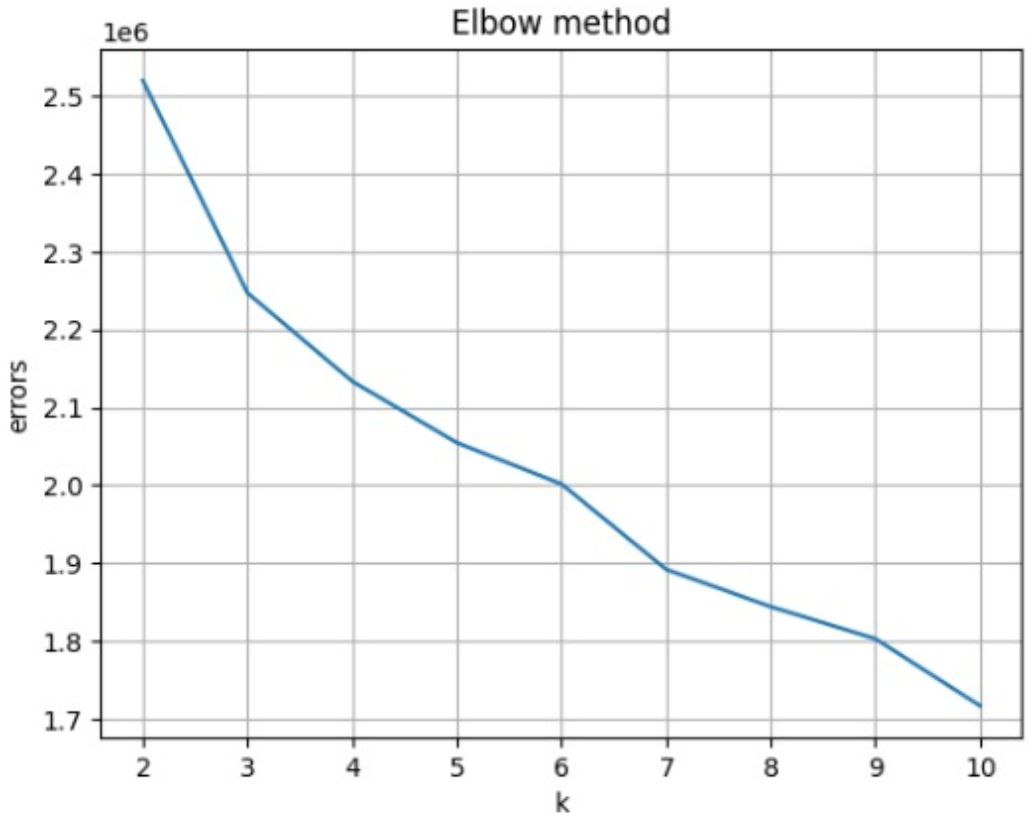
6- Clustering accuracy (KMeans)using Silhouette method, Elbow method

```
plt.plot(k_values, errors)
plt.title('Elbow method')
plt.xlabel('k')
plt.ylabel('errors')
plt.grid(axis='both')
plt.show()
```

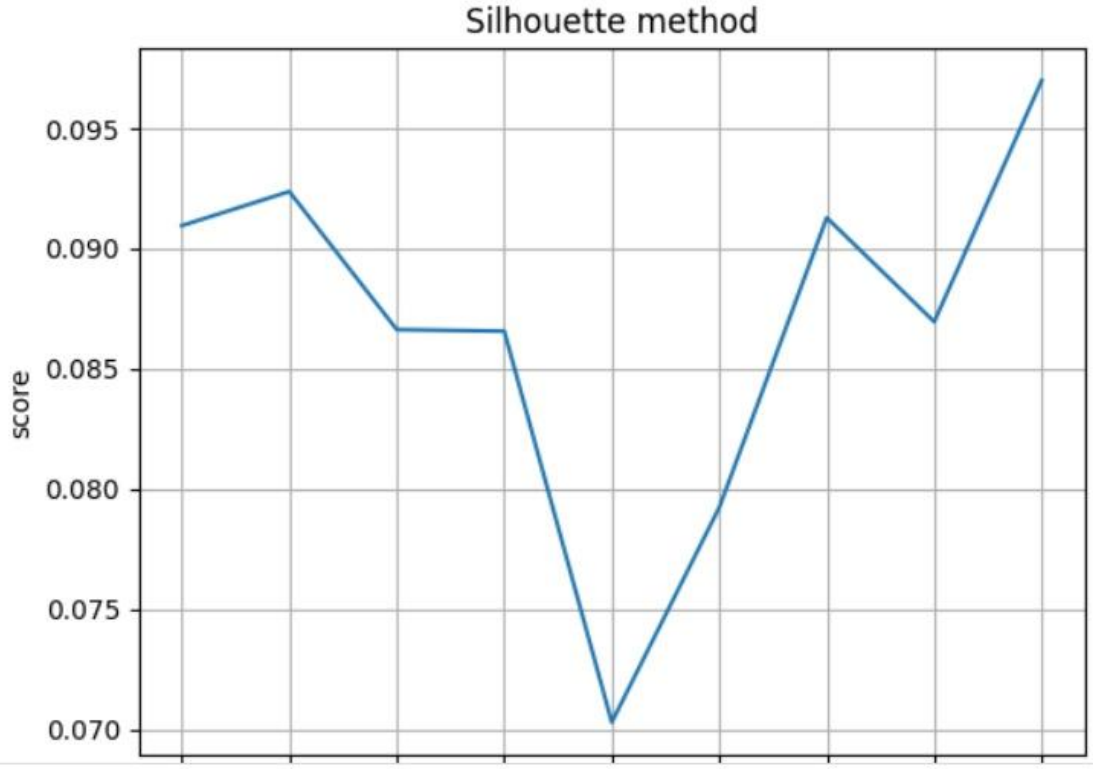


6-curve

```
plt.plot(k_values, errors)
plt.title('Elbow method')
plt.xlabel('k')
plt.ylabel('errors')
plt.grid(axis='both')
plt.show()
```



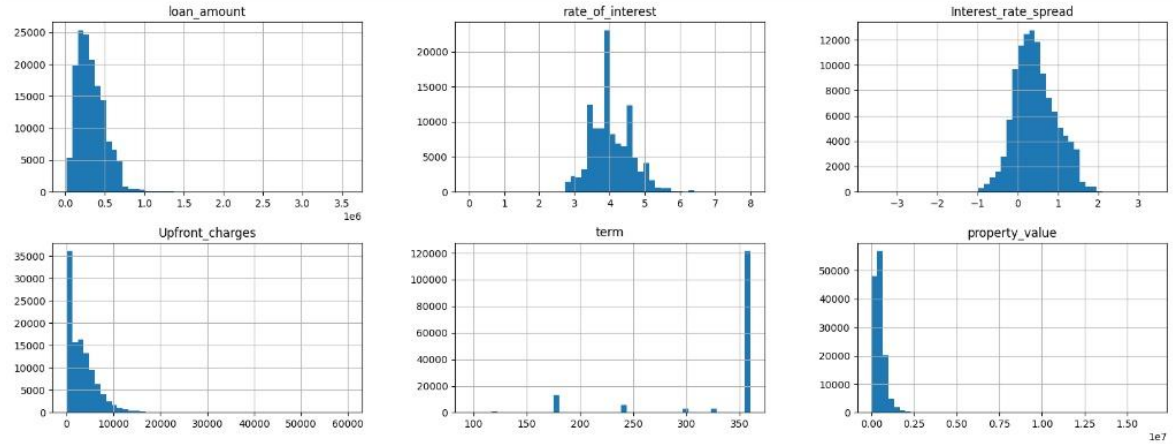
```
plt.plot(k_values, scores)
plt.title('Silhouette method')
plt.xlabel('k')
plt.ylabel('score')
plt.grid(axis='both')
plt.show()
```



Visualization Step:

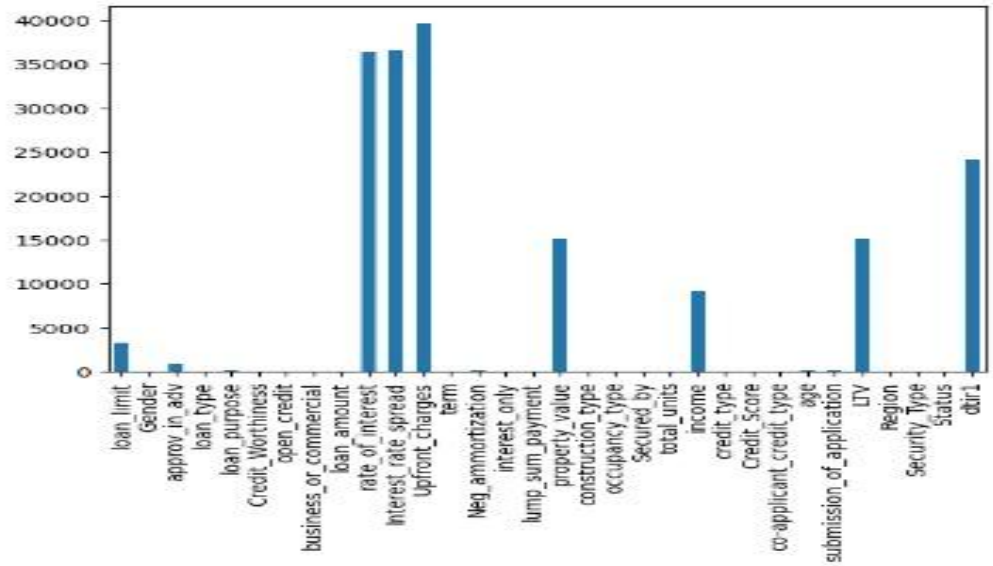
1-histogram

```
# Numerical data histogram to show the distribution  balance of data and ranges
df.hist(bins = 50, figsize = (20, 15))
plt.show()
```



2-bar chart (null values)

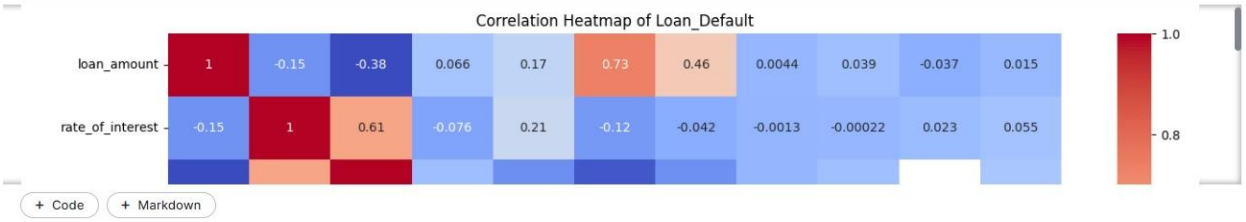
```
df.isna().sum().plot.bar()
plt.show()
```



null values in some features scores approxematly 27%(the maximum) of total data

3-Heat MAP (correlation between numerical columns)

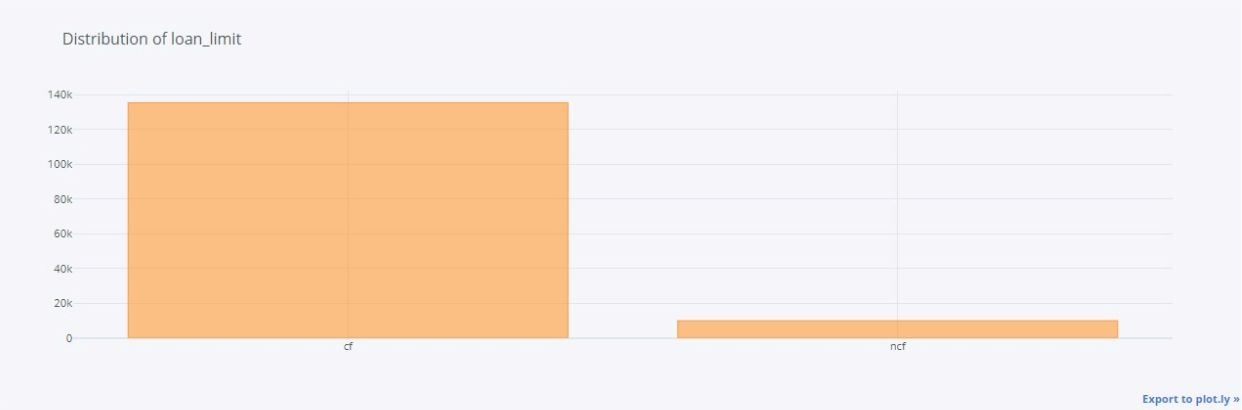
```
# Select only numerical columns for the heatmap
numerical_cols = df.select_dtypes(include=['number'])

# Create a correlation matrix
correlation_matrix = numerical_cols.corr()

# Create a heatmap
plt.figure(figsize=(16, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap of Loan_Default')
plt.show()
```



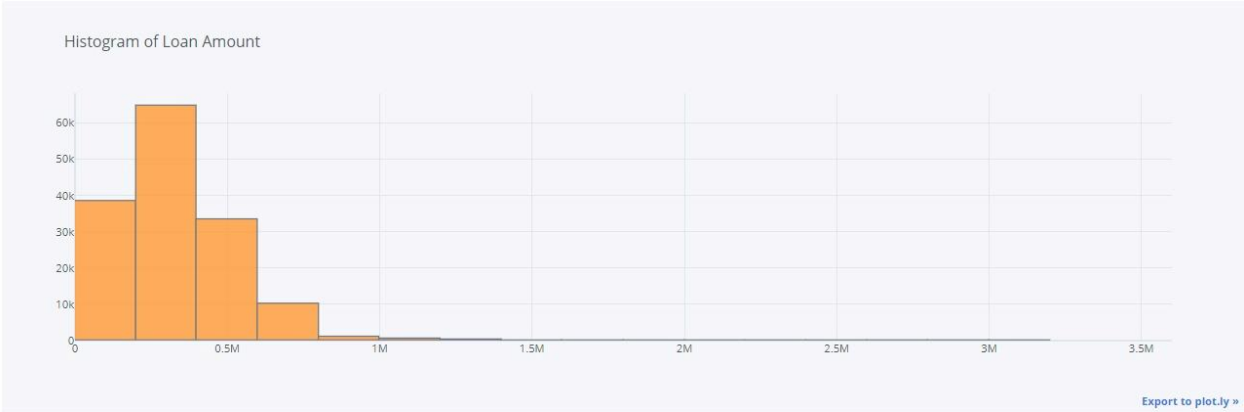
+ Code + Markdown

4-count plot for features

```
df['loan_limit'].value_counts().iplot(kind='bar', title='Distribution of loan_limit')
```

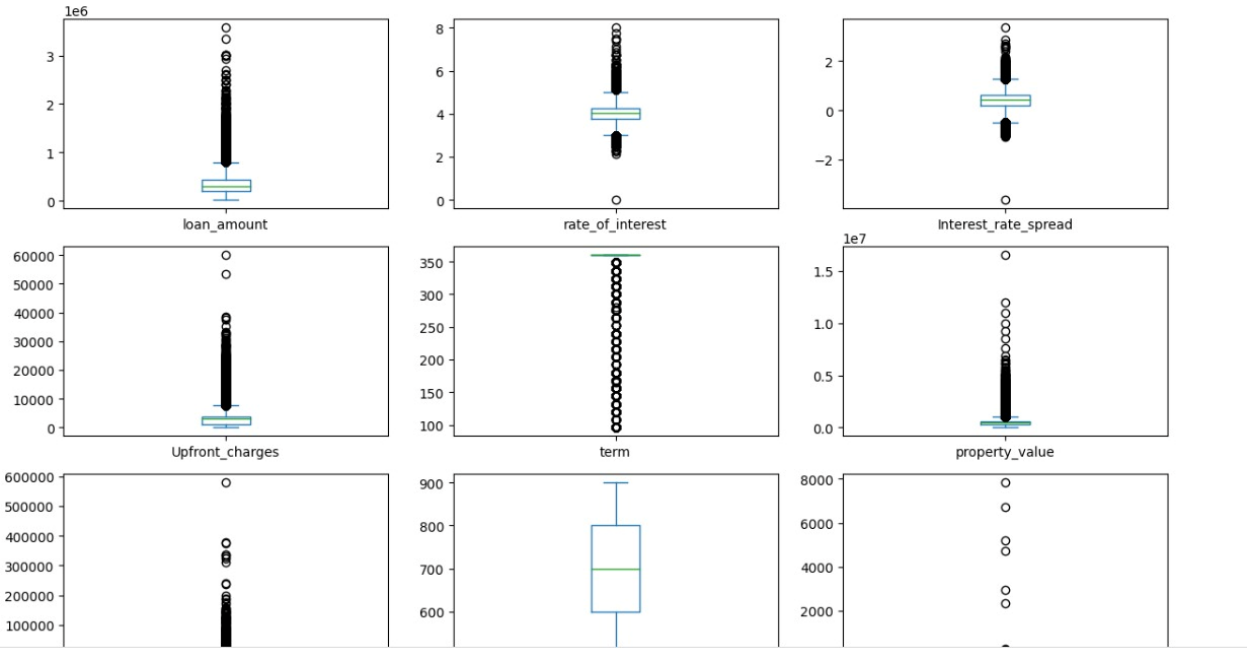


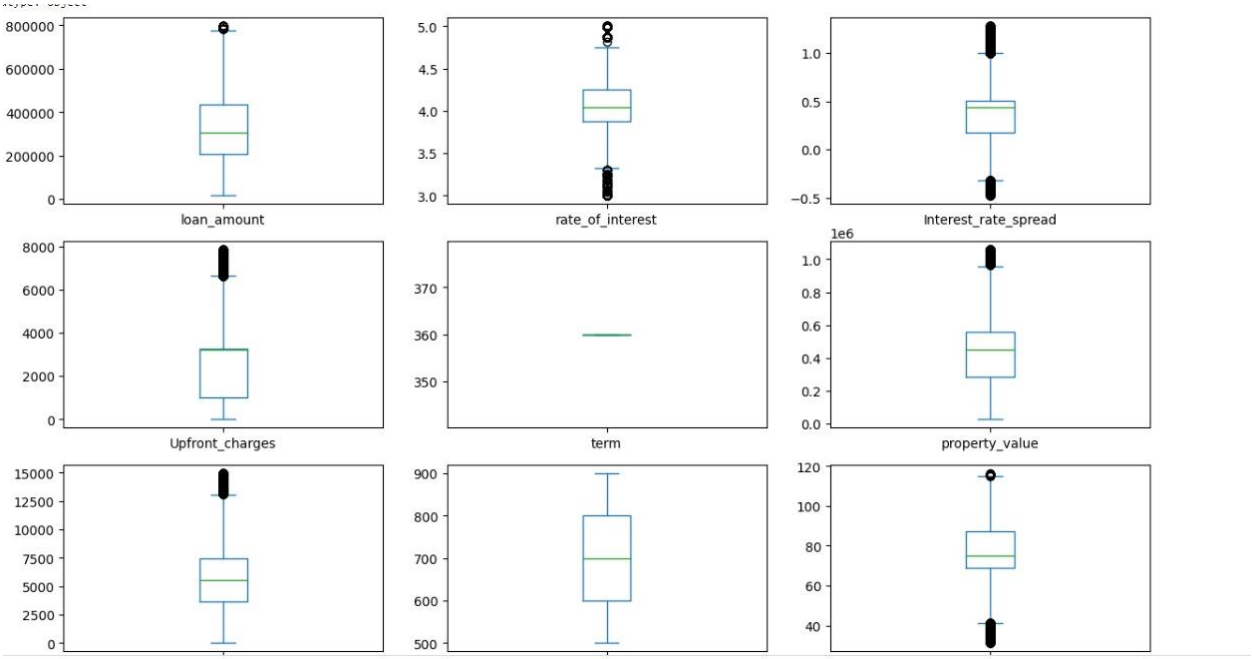
```
df['loan_amount'].iplot(kind='hist', bins=30, title='Histogram of Loan Amount')
```



the most needed loan is is from type1 and thethe amount nearly(200k to 400k)

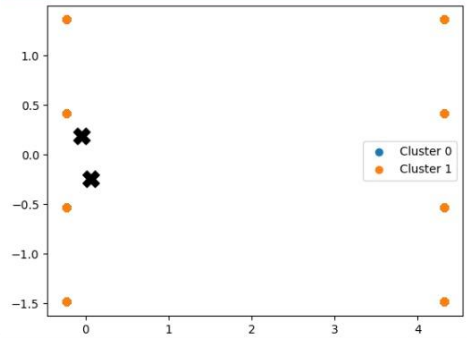
5-box plot for outlier (before, after)removal





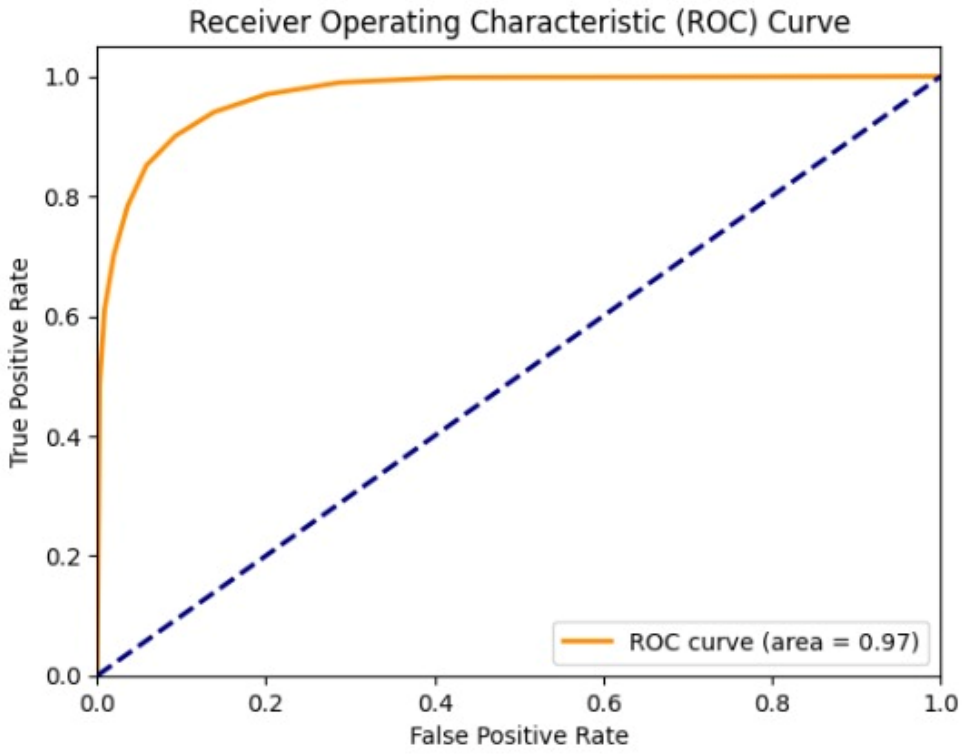

```
# Create and fit KMeans
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# Assign clusters
predicted_labels = kmeans.predict(X)

# Visualize clusters
for i in range(kmeans.n_clusters):
    plt.scatter(X[predicted_labels == i, 0], X[predicted_labels == i, 1], label=f'Cluster {i}')
    plt.scatter(kmeans.cluster_centers_[i][0], kmeans.cluster_centers_[i][1], s=200, marker='X', color='black')  # centroid

plt.legend()
plt.show()
```

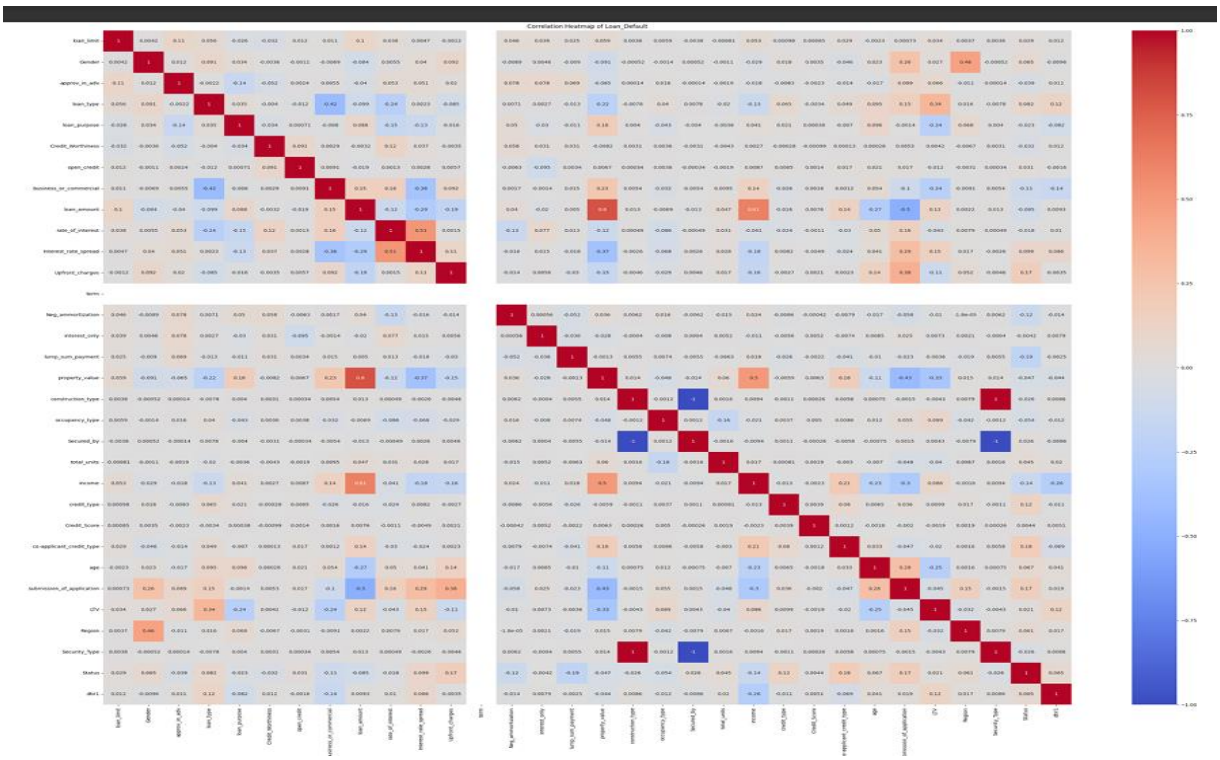
```
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```



8-

---------- KNeighborsClassifier(n_neighbors=10) ----------






FCAI