Notice Find the perfect office for Notebook? Tell us about what's that fits the bill and what's still missing in the office collection. Please rate their updating of Notebook ? Until next version of our extensions.

Hide this alertbox

Data Processing And building machine learning Model

Steps of Preprocessing of data

Step 1: Import necessary libraries, Step 2: Read Data Set, Step 3: Sanity Check of Data Step 4: Exploratory Data Analysis(EDA) Step 5: Missing Value Treatments Step 6: Outliers Treatment, Step 7: Scaledata and Storage Value Imputation Step 8: Normalization Step 9: Encoding of Data

Financial data (Credit risk assessment for loan approval using borrower's financial history)

Type dataset.info() and .describe()

Use .info() to determine the data type of each feature and identify the dataset's overall size and any missing or null values. This is a common application of machine learning particularly classification problems.

Step 1: Import Necessary Library

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

Step 2: Read DataSet

In [3]:
```
dataset_loan = pd.read_csv("loan_prediction.csv")
```

In [4]:
```
# Loan_ID,Gender,Married,Dependents,Education,Self_Employed,ApplicantIncome,CoapplicantIncome,LoanAmount,Loan_Amount_Term,Credit_History,Property_Area,Loan_Status
df = dataset_loan.drop(['Loan_ID'],axis=1)
df.shape

# Display dataset information about the dataset
print("Dimensions of the dataset",df.shape)
print("Characteristics of df",df.sample(2))
print("Feature datatypes\n",df.dtypes)
print("Count of null values",df.isnull().sum())
print("Unique values",df.nunique())
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome',
       'LoanAmount', 'Loan_Amount_Term', 'Credit_History',
       'Property_Area', 'Loan_Status'],
      dtype=object)
```

In [6]:
```
# Count of categorical features
print("Count of Gender",df['Gender'].value_counts())
print("Count of Married",df['Married'].value_counts())
print("Count of Dependents",df['Dependents'].value_counts())
print("Count of Education",df['Education'].value_counts())
print("Count of Self_Employed",df['Self_Employed'].value_counts())
print("Count of Property_Area",df['Property_Area'].value_counts())
print("Count of Credit_History",df['Credit_History'].value_counts())
print("Count of Loan_Amount_Term",df['Loan_Amount_Term'].value_counts())
print("Count of Loan_Status",df['Loan_Status'].value_counts())
```

Out[6]:
```
Count of Gender Male    489
Female    112
NaN        13
Count of Married Yes    398
No     213
Count of Dependents 0    345
1    102
2    101
3+    51
Count of Education Graduate    480
Not Graduate    134
Count of Self_Employed No    500
Yes     82
Count of Property_Area Semiconductor    233
Rural    179
Urban    202
Count of Credit_History 1.0    475
0.0     89
Count of Loan_Amount_Term 360.0    512
180.0     44
480.0     15
300.0     13
84.0       4
240.0      4
120.0      3
36.0       2
60.0       2
12.0       1
Count of Loan_Status Y    422
N    192
```

Step 3: Sanity Check of Data

In [7]:
```
df.head()
df.tail()
```

Out[7]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8312	1083.0	250.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	300.0	360.0	1.0	Urban	N
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiconductor	N

5 rows × 12 columns

In [8]:
```
df.info()
```

Out[8]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             601 non-null    Object
 1   Married            611 non-null    Object
 2   Dependents         599 non-null    Object
 3   Education          614 non-null    Object
 4   Self_Employed      582 non-null    Object
 5   ApplicantIncome    614 non-null    Int64
 6   CoapplicantIncome  614 non-null    Float64
 7   LoanAmount         592 non-null    Float64
 8   Loan_Amount_Term   600 non-null    Float64
 9   Credit_History     564 non-null    Float64
 10  Property_Area      614 non-null    Object
 11  Loan_Status        614 non-null    Object
dtype: Int64(1), Float64(4), Object(7)
memory usage: 57.7 KB
```

In [9]:
```
df[df.duplicated].shape
```

Out[9]:
```
(0, 12)
```

In [10]:
```
df.shape
```

Out[10]:
```
(614, 12)
```

Step 4: Exploratory Data Analysis

In [11]:
```
df.describe()
```

Out[11]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

5 rows × 7 columns

In [12]:
```
# Numerical features the distribution
df.hist(figsize=(15,15))
plt.show()
```

In [13]:
```
# Categorical features the distribution
df.hist(figsize=(15,15))
plt.show()
```

In [14]:
```
# Boxplot to understand the distribution
# Numerical features
df.plot(kind='box',figsize=(15,10))
```

In [15]:
```
# Boxplot to understand the distribution
# Numerical features
df.plot(kind='box',figsize=(15,10))
```

In [16]:
```
# Boxplot to understand the relationship
# Numerical features VS Target
df.boxplot(column=['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History'],
           by=['Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Property_Area', 'Loan_Status'],
           figsize=(15,15))
```

In [17]:
```
# Count of Target
df['Loan_Status'].value_counts()
```

Out[17]:
```
Loan_Status
Y    422
N    192
Name: count, dtype: Int64
```

In [18]:
```
# Count of Target
df['Loan_Status'].value_counts()
```

Out[18]:
```
Loan_Status
Y    422
N    192
Name: count, dtype: Int64
```

In [19]:
```
# Count of Target
df['Loan_Status'].value_counts()
```

Out[19]:
```
Loan_Status
Y    422
N    192
Name: count, dtype: Int64
```

In

```
print("Final after cleaning (tuf_clean):")
tuf_clean.shape
```

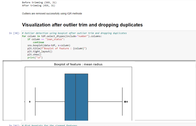
```
Final after cleaning (tuf_clean):
(3316, 36)
```

```
# Check class distribution after cleaning
print("Class distribution after cleaning:")
print(tuf_clean['class'].value_counts())
print(tuf_clean['class'].value_counts(normalized=True))
```

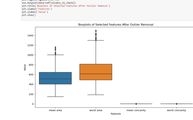
```
Class distribution after cleaning:
class
Positive    2062
Negative    1254
Name: count, dtype: int64
class
Positive    0.621
Negative    0.379
Name: proportion, dtype: float64
```

Visualization after outlier trim and dropping duplicates

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[:4]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show()
```

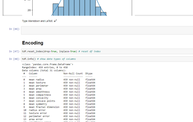



```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[4:8]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```




```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[8:12]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[12:16]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```




```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[16:20]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[20:24]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[24:28]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[28:32]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[32:36]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[36:40]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[40:44]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[44:48]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[48:52]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[52:56]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[56:60]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
# Boxplot comparison after removal after outlier trim and dropping duplicates
for col in numerical_features[60:64]:
    plt.figure(figsize=(8, 4))
    sns.boxplot(
        x=tuf_clean[col],
        y=tuf_clean['class'],
        palette=['#2ecc71', '#e74c3c'],
        order=['Negative', 'Positive']
    )
    plt.title(f'Boxplot of Feature: {col}')
    plt.xlabel(col)
    plt.ylabel('Class')
    plt.show
```

```
print("Final shape:")
tuf_clean.shape
```

```
Final shape:
(3316, 36)
```

```
tuf_clean.info()
```

```
```