In [101…
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, KFold
from sklearn.linear_model import LinearRegression
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from statsmodels.formula.api import ols
```

In [102…
```
train_1 = pd.read_csv("train_1.csv", index_col = 0)
train_2 = pd.read_csv("train_2.csv", index_col = 0)
train = pd.concat([train_1 , train_2], sort=True)
print(train.shape)
```

```
(614, 12)
```

In [103…
```
print(train.info())
print(train.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 614 entries, LP001002 to LP002990
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   ApplicantIncome    614 non-null    int64  
 1   CoapplicantIncome  614 non-null    float64
 2   Credit_History     564 non-null    float64
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Gender             601 non-null    object 
 6   LoanAmount         592 non-null    float64
 7   Loan_Amount_Term   600 non-null    float64
 8   Loan_Status        614 non-null    object 
 9   Married            611 non-null    object 
 10  Property_Area      614 non-null    object 
 11  Self_Employed      582 non-null    object 
dtypes: float64(4), int64(1), object(7)
memory usage: 62.4+ KB
None
       ApplicantIncome  CoapplicantIncome  Credit_History  LoanAmount  \
count       614.000000         614.000000      564.000000  592.000000   
mean       5403.459283        1621.245798        0.842199  146.412162   
std        6109.041673        2926.248369        0.364878   85.587325   
min         150.000000           0.000000        0.000000    9.000000   
25%        2877.500000           0.000000        1.000000  100.000000   
50%        3812.500000        1188.500000        1.000000  128.000000   
75%        5795.000000        2297.250000        1.000000  168.000000   
max       81000.000000       41667.000000        1.000000  700.000000   

       Loan_Amount_Term  
count         600.00000  
mean          342.00000  
std            65.12041  
min            12.00000  
25%           360.00000  
50%           360.00000  
75%           360.00000  
max           480.00000  
```

In [104…
```
print(train.isna().sum())
```

```
ApplicantIncome       0
CoapplicantIncome     0
Credit_History       50
Dependents           15
Education             0
Gender               13
LoanAmount           22
Loan_Amount_Term     14
Loan_Status           0
Married               3
Property_Area         0
Self_Employed        32
dtype: int64
```

In [105…
```
threshold = len (train) * 0.05
print (threshold)
```

```
30.700000000000003
```

In [106…
```
cols_to_drop = train.columns [train.isna().sum() <= threshold]
print (cols_to_drop)
train.dropna(subset=cols_to_drop, inplace=True)
```

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

In [107…
```
cols_with_missing_valves = train.columns[train.isna().sum() > 0]
print (cols_with_missing_valves)
```

```
Index(['Credit_History', 'Self_Employed'], dtype='object')
```

In [108…
```
for col in cols_with_missing_valves[:-1]:
    train[col].fillna(train [col].mode () [0])
```

This is because some rows may have contained missing values for our subset columns as well as salary, so they were dropped.

In [109…
```
print(train.isna().sum())
```

```
ApplicantIncome       0
CoapplicantIncome     0
Credit_History       48
Dependents            0
Education             0
Gender                0
LoanAmount            0
Loan_Amount_Term      0
Loan_Status           0
Married               0
Property_Area         0
Self_Employed        30
dtype: int64
```

In [110…
```
train_dict = train.groupby ("Education")["Credit_History"].median ().to_dict
print (train_dict)
```

```
{'Graduate': 1.0, 'Not Graduate': 1.0}
```

In [111…
```
train["Credit_History"] = train["Credit_History"].fillna(train["Education"].
print(train.isna().sum())
```

```
ApplicantIncome       0
CoapplicantIncome     0
Credit_History        0
Dependents            0
Education             0
Gender                0
LoanAmount            0
Loan_Amount_Term      0
Loan_Status           0
Married               0
Property_Area         0
Self_Employed        30
dtype: int64
```

In [112…
```
corr_App_Loan = train['ApplicantIncome'].corr(train["LoanAmount"])
print(corr_App_Loan)
```

```
0.529727803407427
```

In [113…
```
sns.heatmap(x = "ApplicantIncome" , y = "LoanAmount", data = train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_33181/1890650366.
py in <module>
----> 1 sns.heatmap(x = "ApplicantIncome" , y = "LoanAmount", data = train)

~/opt/anaconda3/lib/python3.9/site-packages/seaborn/matrix.py in heatmap(dat
a, vmin, vmax, cmap, center, robust, annot, fmt, annot_kws, linewidths, line
color, cbar, cbar_kws, cbar_ax, square, xticklabels, yticklabels, mask, ax,
**kwargs)
    444     """
    445     # Initialize the plotter object
--> 446     plotter = _HeatMapper(data, vmin, vmax, cmap, center, robust, an
not, fmt,
    447                           annot_kws, cbar, cbar_kws, xticklabels,
    448                           yticklabels, mask)

~/opt/anaconda3/lib/python3.9/site-packages/seaborn/matrix.py in __init__(se
lf, data, vmin, vmax, cmap, center, robust, annot, fmt, annot_kws, cbar, cba
r_kws, xticklabels, yticklabels, mask)
    161
    162         # Determine good default values for the colormapping
--> 163         self._determine_cmap_params(plot_data, vmin, vmax,
    164                                     cmap, center, robust)
    165

~/opt/anaconda3/lib/python3.9/site-packages/seaborn/matrix.py in _determine_
cmap_params(self, plot_data, vmin, vmax, cmap, center, robust)
    195
    196         # plot_data is a np.ma.array instance
--> 197         calc_data = plot_data.astype(float).filled(np.nan)
    198         if vmin is None:
    199             if robust:

ValueError: could not convert string to float: '3+'
```

In [ ]:
```
sns.scatterplot(data = )
```