Machine Learning With Python

Internship Project Report

LOAN AMOUNT PREDICTION

At

IC SOLUTIONS

Submitted by:-

Name	USN
ADARSH N	1GG18CS002
adarshkumar738291@gmail.com	
VINAYAK S BAGANNANAVAR	1GG18CS043
bagannavarvinayak123@gmail.com	

COURSE INSTRUCTOR:- ABHISHEK C

ACKNOWLEDGMENT :

We would like to thank both the Take It Easy Engineering team and IC Solutions for offering this internship to explore the field of machine learning in python with a great mentor.

We also like to express our sincere gratitude to Visvesvaraya Technological University for giving us this great opportunity.

We are grateful to have the most cooperating and helping mentor Mr Abhishek C Sir.

Being with us throughout our project and providing all required guidelines and materials which were not only helpful for our project but also to our future.

We sincerely thank our parents and friends for supporting and motivating us during our hard times in the project.

Last but not the least, we would like to thank all the people who knowingly or unknowingly helped us to achieve this project.

ABSTRACT :

With the enhancement in the banking sector, lots of people are applying for bank loans but the bank has its limited assets which it has to grant to limited people only, so finding out to whom the loan can be granted and the loan amount that will be a safer option for the bank is a typical process. Even after taking a lot of precautions and analyzing the mortgage applicant information, the mortgage approval choices are not continually correct. There is a need for automation of this system so that loan approval is much less risky and ensures less loss for the banks. Since it is a major activity of the banks, to identify whether a loan of the desired amount should be sanctioned to the applicant or not, Computer Science is capable of making such a system using Machine Learning, which can make these tough decisions quickly and accurately.

The dataset containing information of several loan applicants is divided into training and testing data and it is fed to an algorithm. The Algorithm trains the training data and fits it in the model, then it is tested using testing data and predicts. The main objective of this project is to predict the loan amount to a particular person based on the details. Machine Learning algorithms like Linear Regression model, Decision Tree Regressor model, Random Forest Regressor model, Bayesian Ridge Regression model, Gradient Boosting Regressor model, XGB Regression model, Artificial neural networks and so on. These are the efficient algorithms that are followed for data analysis and prediction making.

The system will look into some basic information of the applicant such as his/her Loan Requested amount, Property value, Income, Credit History, etc., and after analyzing all this information, using visualization and Machine Learning algorithms, it will predict the loan amount that can be sanctioned.

ABOUT THE COMPANY :

ICS is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At ICS, we believe that service and quality is the key to success.

We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

Development - We develop responsive, functional and super-fast websites. We keep User Experience in mind while creating websites. A website should load quickly and should be accessible even on a small view-port and slow internet connection.

Mobile Application - We offer a wide range of professional android, iOS & Hybrid app development services for our global clients, from a start-up to a large enterprise.

Design - We offer professional Graphic design, Brochure design & Logo design. We are experts in crafting visual content to convey the right message to the customers.

Consultancy - We are here to provide you with expert advice on your design and development requirements.

Videos - We create a polished professional video that impresses your audience.

INDEX :

INTRODUCTION :

The Internship was in the specialization of Python with Machine Learning, which was offered by IC Solutions for four weeks, where the first nine days were learning sessions and the rest was for the project. Our mentor Mr Abhishek C is a highly skilled professional Machine Learning Engineer who supported us to understand from scratch to explore the field of machine learning with python. He started with the basics of the python programming language, exploring the libraries like NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, Tensorflow and the concepts with examples during the coding of machine learning models, artificial neural networks, etc.

Machine Learning is an interesting, evolving and future of technology as it is a key to build advanced automation systems. This internship was the key to explore in the field of Machine Learning. The concepts and coding experience we earned in the internship is of great value and is a part of our career growth opportunity in the future.

This is a regression project as we predict the loan amount, it has three phases which are Exploratory data analysis, preparing machine learning models, and analyzing the output. Phase one, Exploratory data analysis deals with the preprocessing methods like understanding the data by plotting graphs, charts, maps, etc., data cleaning, adding dummies to the null values, standardizing the data and splitting training and testing data. Phase two, Preparing machine learning models such as Linear Regression Model, Random Forest Regression Model, Decision Tree Regression Model, Bayesian Ridge Model, K Neighbors Regression Model, Artificial Neural Networks and a few other regression models. Phase three, analyze the accuracy of all the models and choose the best model.

Problem Statement :

The bank wants to automate the process of sanctioning the loan amount in real-time based on details provided by the loan applicant's, as the process of calculating the loan amount is a difficult task and requires lots of dedication and hard work. Using the latest machine learning techniques we can find the best algorithm or model that predicts the loan amount to an accurate level. We are given a dataset consisting of 30,000 customer's loan application details with all the required information and the loan sanctioned amount. We need to use the dataset provided to train the machine learning model and predict the loan amount for new customer loan details. A minimum of three Machine Learning models should be trained and tested using the given dataset.

Objectives:-

- Identifying the type of problem that can be solved. Regression is the best type to predict the loan amount to be sanctioned.
- Using the exploratory data analysing techniques to analyse the data by plotting graphs, to find the relation between the data features and preprocess the data by removing unwanted columns, clearing null values and dividing the dataset into training and testing data.
- Build and fit the training data to train the Machine Learning models, to predict the loan amount to be sanctioned.
- Repeat the above process to all Machine Learning Models.
- Analyse the best prediction by testing the Machine Learning model based on the r2 score.

Requirement Specification :

Hardware Specification:-

- CPU: 2 x 64-bit, 2.8 GHz, 8.00 GT/s CPUs or better.
- Memory: RAM size of 4GB or greater.
- Storage: Hard disk of 100GB or greater.

Software Specification:-

- Operating System: Windows 7/8/10/11, macOS or Linux
- Compiler: Python version greater than 3.0
- Libraries: NumPy, Pandas, Scikit-Learn, Seaborn, Matplotlib and TensorFlow.
- Platform: Anaconda
- Editor: Jupyter

Virtual Platforms:-

These are a few of many virtual platforms which help in the execution of the algorithm in any browser.

- Google Colab
- Kaggle
- OpenML
- CloudXLab

Exploratory Data Analysis :

This part contains both Data Cleaning and Data Visualization.

1) Details of the given dataset.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 30000 entries, 0 to 29999

Data columns (total 24 columns):

#	Column	Non-Null Count	Dtype
0	Customer ID	30000 non-null	object
1	Name	30000 non-null	object
2	Gender	29947 non-null	object
3	Age	30000 non-null	int64
4	Income (USD)	25424 non-null	float64
5	Income Stability	28317 non-null	object
6	Profession	30000 non-null	object
7	Type of Employment	22730 non-null	object
8	Location	30000 non-null	object
9	Loan Amount Request (USD)	30000 non-null	float64
10	Current Loan Expenses (USD)	29828 non-null	float64
11	Expense Type 1	30000 non-null	object
12	Expense Type 2	30000 non-null	object
13	Dependents	27507 non-null	float64
14	Credit Score	28297 non-null	float64
15	No. of Defaults	30000 non-null	int64

```
 16  Has Active Credit Card       28434 non-null  object
 17  Property ID                  30000 non-null  int64
 18  Property Age                 25150 non-null  float64
 19  Property Type                30000 non-null  int64
 20  Property Location            29644 non-null  object
 21  Co-Applicant                 30000 non-null  int64
 22  Property Price               30000 non-null  float64
 23  Loan Sanction Amount (USD)   29660 non-null  float64
dtypes: float64(8), int64(5), object(11)
memory usage: 5.5+ MB
```

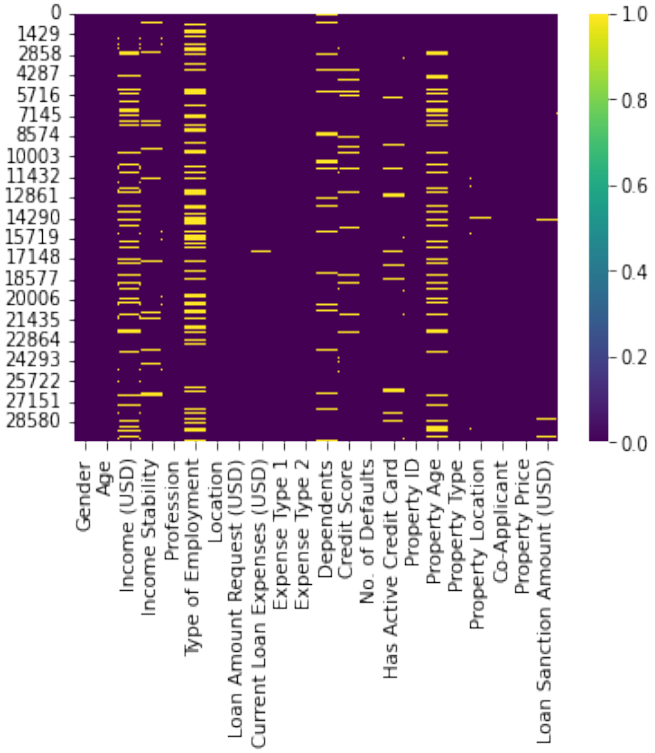
2) Removing the unique features which are unwanted.

Code:

```
loan.drop(["Customer ID","Name", "Property ID"], axis=1, inplace=True)
```

3) Finding null values Using Seaborn heatmap.

Code: sns.heatmap(loan.isnull(),cmap='viridis')

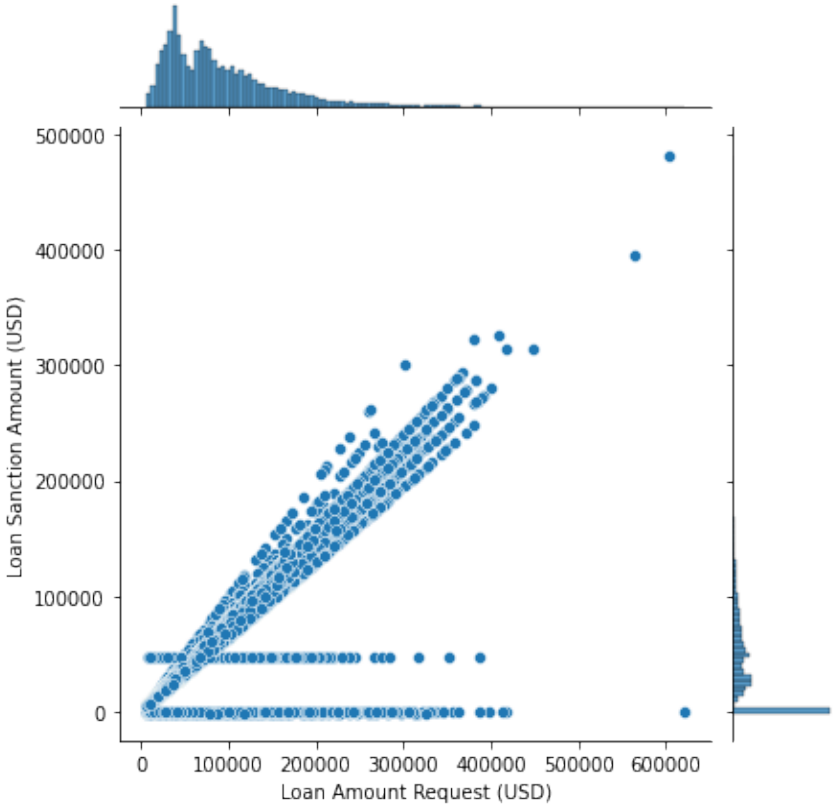


- All the null values are indicated in yellow colour, the purple colour indicates that there is no null value. As there are more null values in the dataset it is better to fill the null values first.
- Income (USD), Current Loan Expenses (USD), Dependents, Credit Score, Property Age and Loan Sanction Amount(USD) are the numeric type features that have null values, which can be replaced by the mean of that particular feature.
- Income Stability, Type of Employment, Has Active Credit Card and Property Location are object type features that have null values, they should be encoded (converting the object to the numeric data type) and replaced by the mode of that particular feature.

4) Plotting joint plot from seaborn library on Loan Amount Request (USD) vs Loan Sanction Amount (USD)

Code:

```
sns.jointplot(x='Loan Amount Request (USD)',y='Loan Sanction Amount (USD)',data=loan)
```

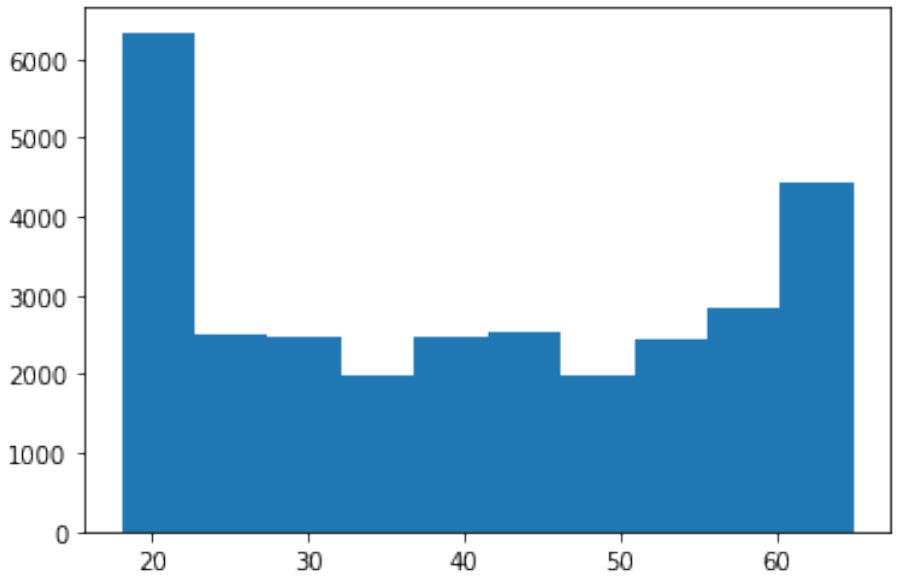


- They look like they are directly proportional.
- Loan Sanction Amount(USD) remained 0, for some Loan Request Amount, it means that their loan was not approved.

5) This plot is to see the range of ages of people applying for loans.

Code:

```
plt.hist(loan['Age'])
```

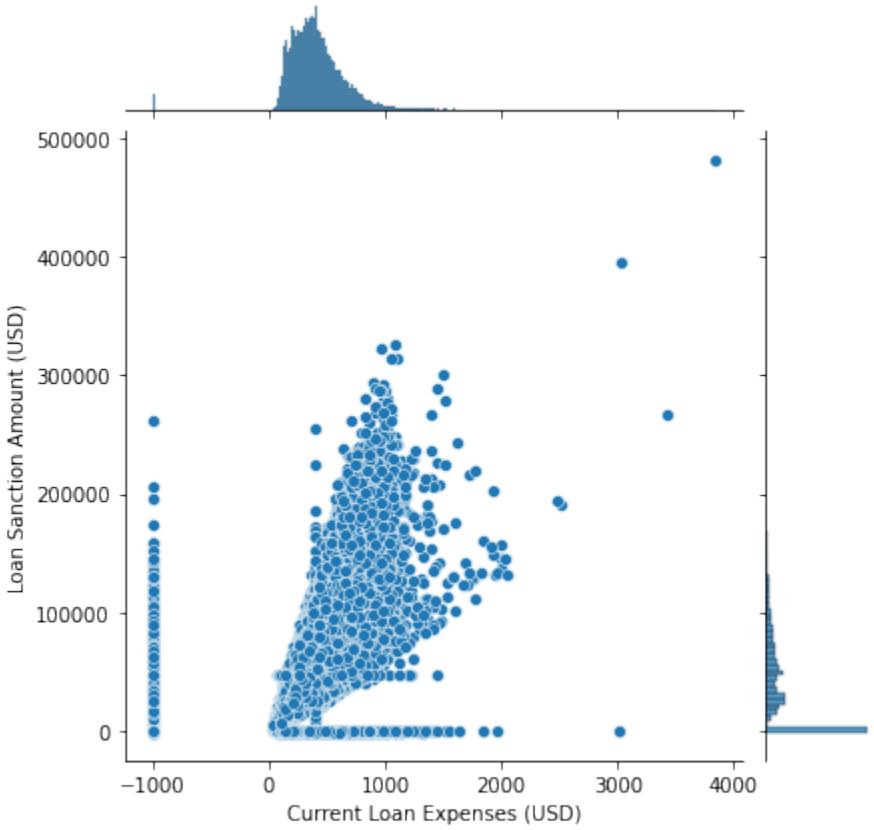


- It seems that most of the people are aged 20.
- And people with age 60 are also more in number.

6) Plotting joint plot from seabron library on Current Loan Expenses (USD) vs Loan Sanction Amount (USD).

Code:

sns.jointplot(x='Current Loan Expenses (USD)',y='Loan Sanction Amount (USD)',data=loan).

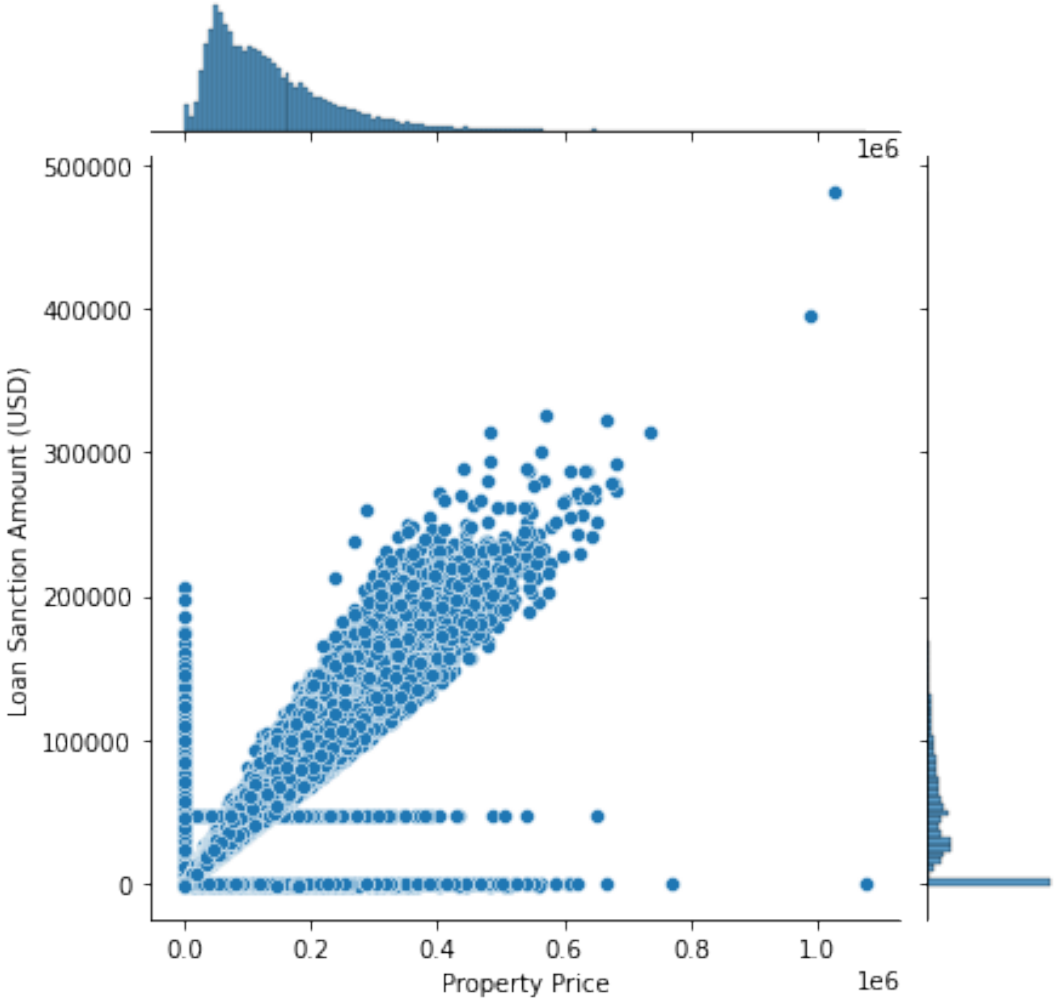


- They seem to be directly proportional.
- Most of the loan expenses lie between 0 to 2000.

7) Plotting joint plot from seabron library on Property Price vs Loan Sanction Amount (USD)

Code:

```
sns.jointplot(x='PropertyPrice',y='Loan Sanction Amount (USD)',data=loan)
```

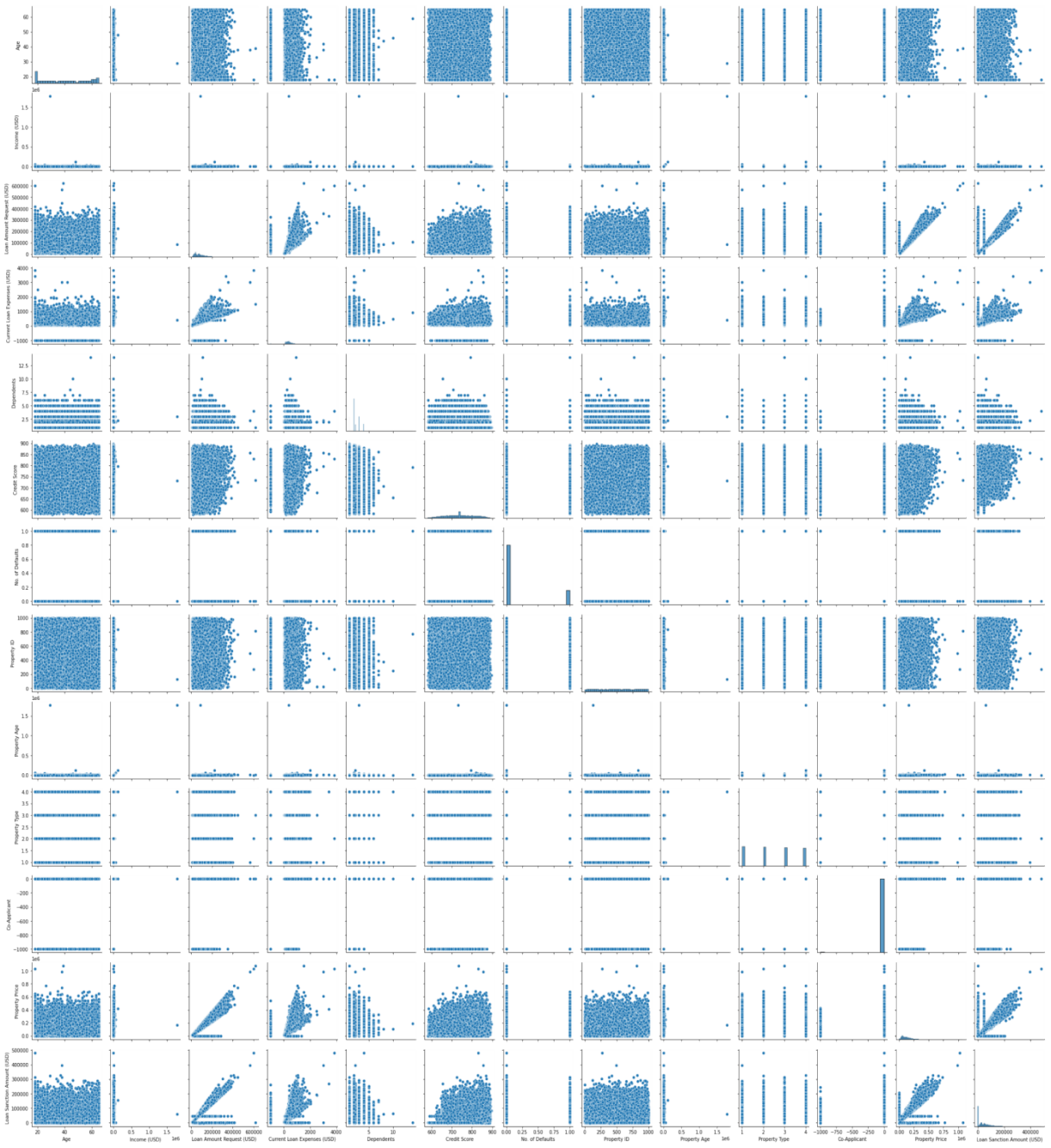


- They look like they are directly proportional.
- Loan Sanction Amount is more in number for Property Price 0.2 to 0.4.

8) Plotting pairplot to visualize the relation between each features

Code:

sns.pairplot(loan)



- While more features are directly proportional to each other.
- So it would be better to start with linear regression.

9) Filling the null values of numeric data type by the mean of the feature.

Code:

```
n_cols = loan[["Income (USD)", "No. of Defaults", "Property Type", "Current Loan Expenses (USD)", "Co-Applicant", "Property Price", "Dependents", "Credit Score", "Property Age", "Loan Amount Request (USD)", "Loan Sanction Amount (USD)"]]
for i in n_cols:
  loan[i].fillna(loan[i].mean(axis=0), inplace=True)
```

10) Filling the null values of object data type by the mode of the feature.

Code:

```
cols = loan[["Gender", "Property Location", "Income Stability", "Has Active Credit Card", "Type of Employment", "Has Active Credit Card", "Expense Type 1", "Expense Type 2"]]
for  i in cols:
      loan[i].fillna(loan[i].mode().iloc[0], inplace=True)
```

11) Converting all the object data types into numeric using the ordinal encoding method.

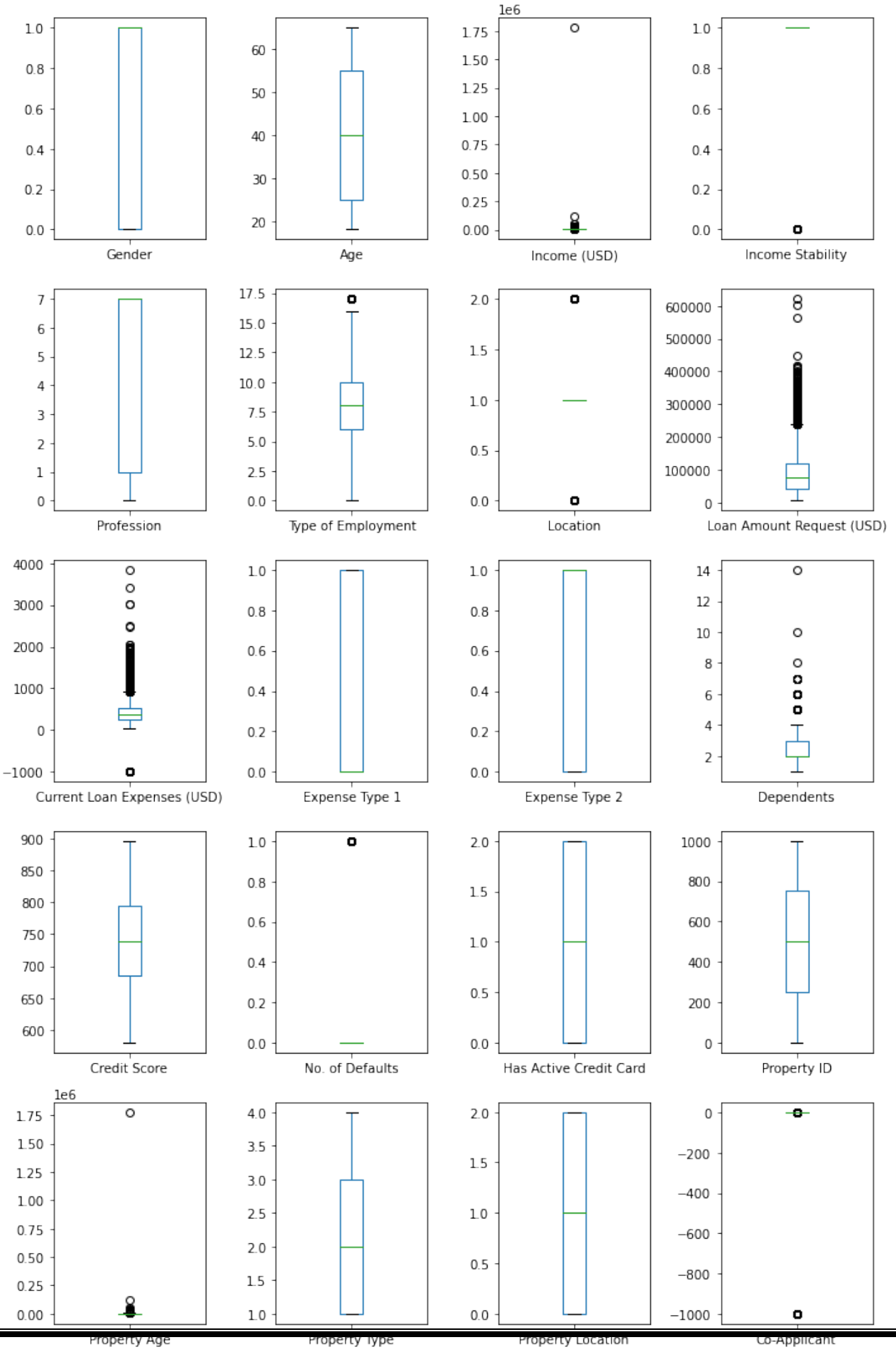
Code:

```
from sklearn.preprocessing import OrdinalEncoder
ord_enc = OrdinalEncoder()
loan[["Gender", 'Income Stability', 'Profession', 'Location', 'Expense Type 1', 'Expense Type 2', 'Has Active Credit Card', 'Type of Employment', 'Property Location']] = ord_enc.fit_transform(loan[["Gender", 'Income Stability', 'Profession', 'Location', 'Expense Type 1', 'Expense Type 2', 'Has Active Credit Card', 'Type of Employment', 'Property Location']])
```

12) BoxPlot

Code:

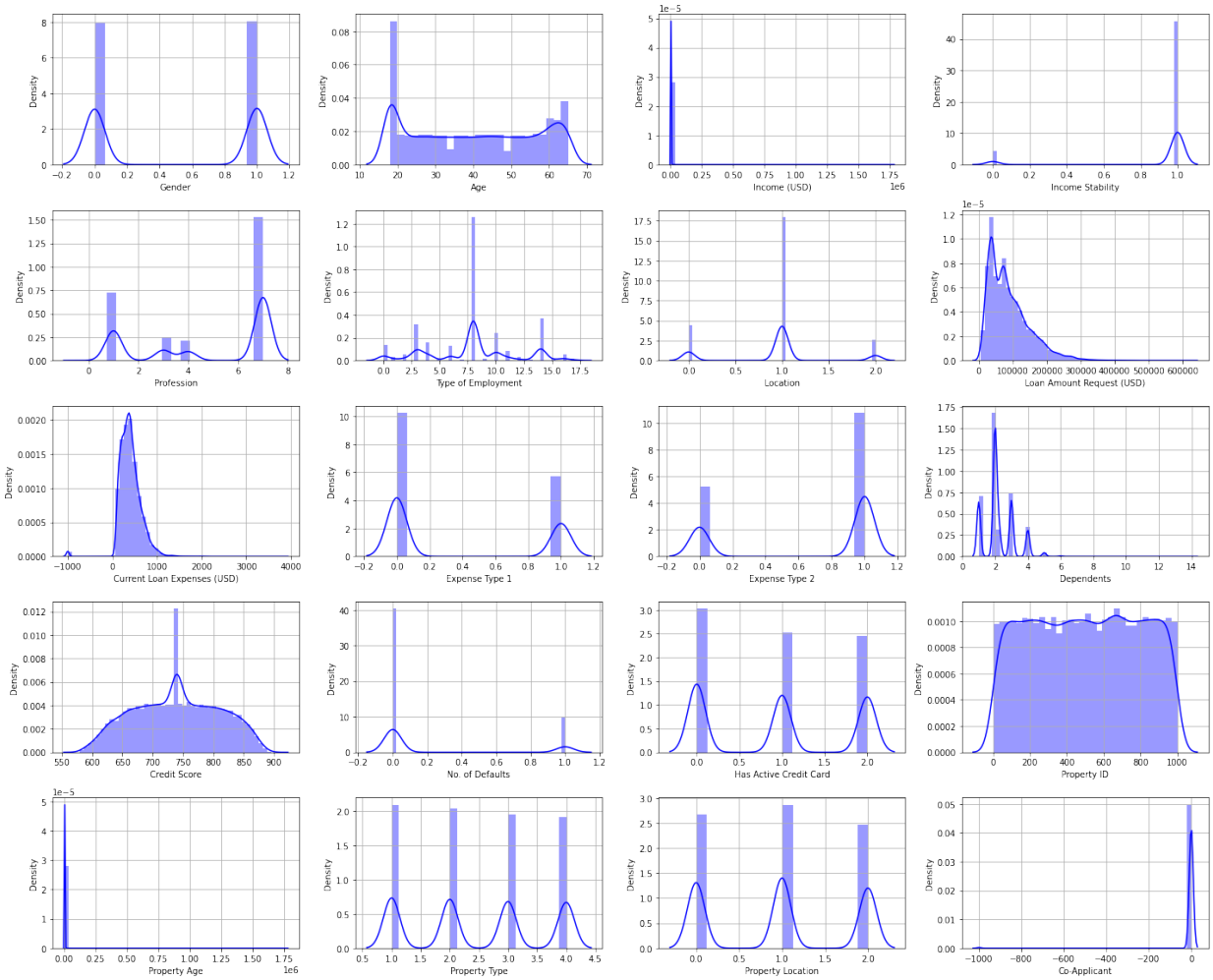
```
plt.figure(figsize=(10,15))
try:
    for i, col in enumerate(list(loan.columns.values)):
        plt.subplot(5,4,i+1 )
        loan.boxplot(col)
        plt.grid()
        plt.tight_layout()
except ValueError:
    pass
```


Gender
Age
1e6
Income (USD)
Income Stability
Profession
Type of Employment
Location
Loan Amount Request (USD)
Current Loan Expenses (USD)
Expense Type 1
Expense Type 2
Dependents
Credit Score
No. of Defaults
Has Active Credit Card
Property ID
1e6
Property Age
Property Type
Property Location
Co-Applicant

13) Subplot

Code:

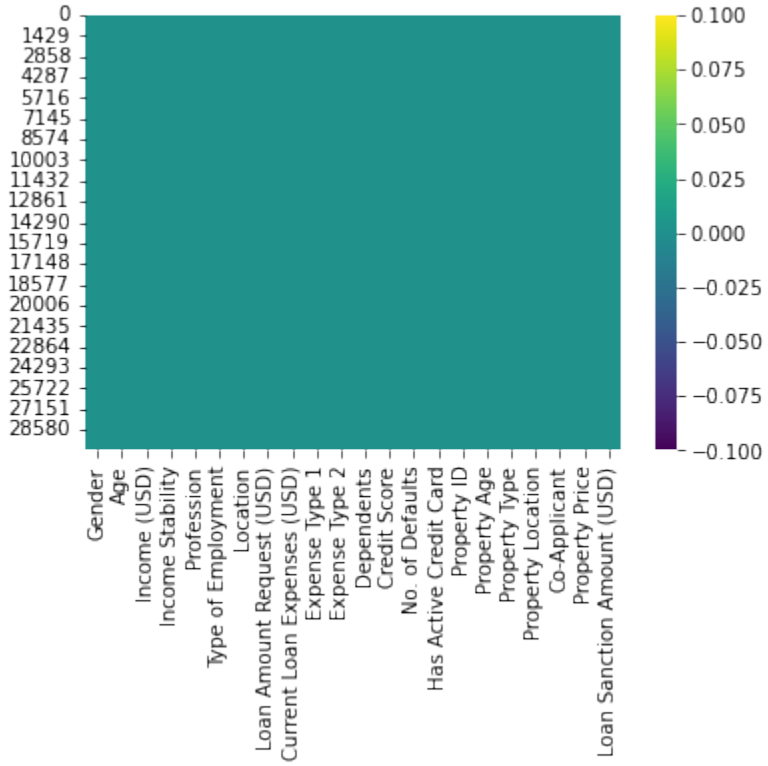
```
plt.figure(figsize=(20,16))
try:
    for i,col in enumerate(list(loan.columns.values)):
    plt.subplot(5,4,i+1)
        sns.distplot(loan[col], color='b', kde=True, label='data')
    plt.grid()
    plt.tight_layout()
except ValueError:
    pass
```



14) Cross-checking with heatmap.

Code:

```
sns.heatmap(loan.isnull(),cmap='viridis')
```

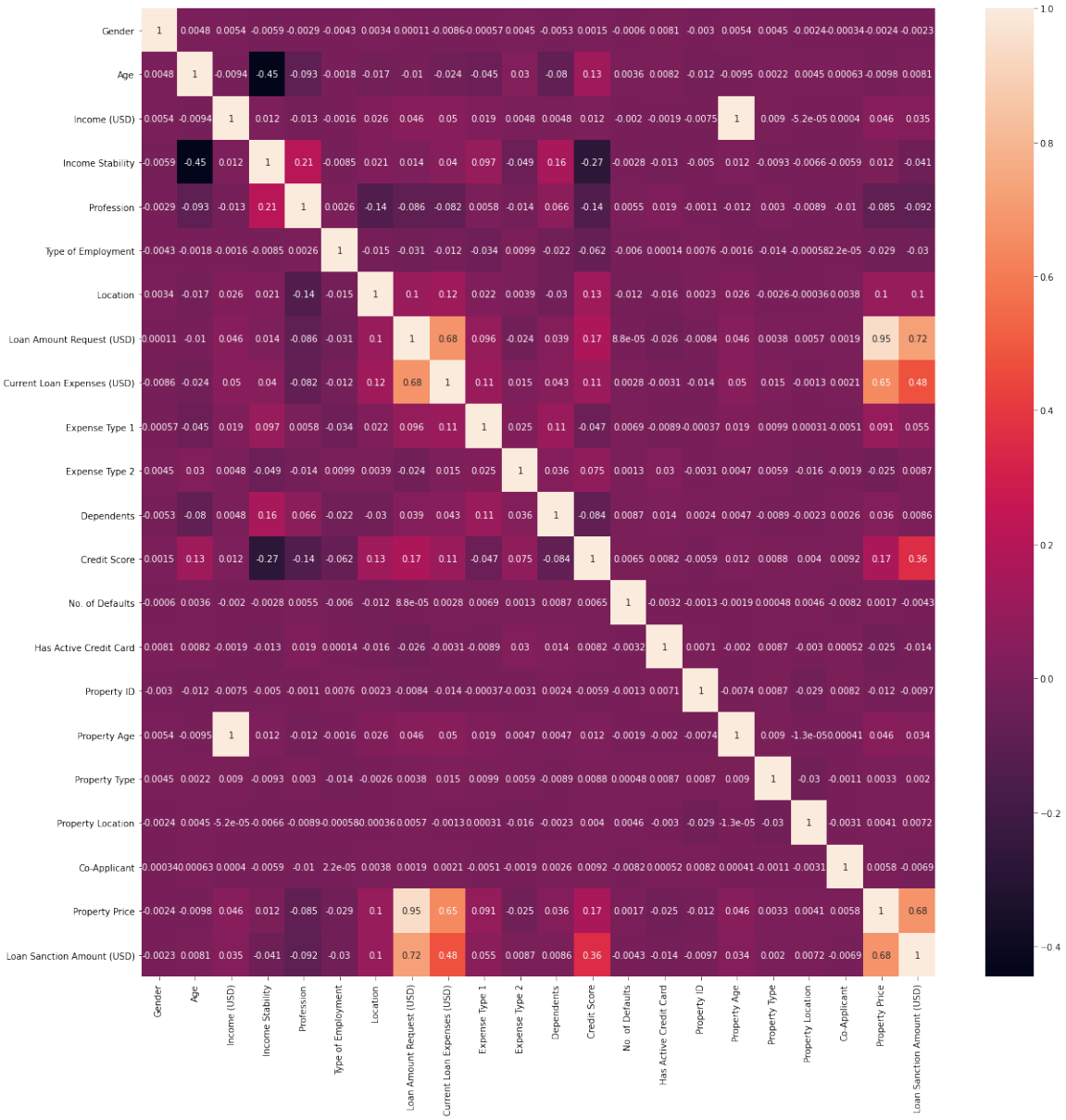


- This plot shows the null values in the dataset.
- The numbers from -1 to 1 indicate the number of null values.
- The number 0 colour (Blue) indicates that there are no null values.

15) Plotting heatmap to see the relation between each features

Code:

```
cm = loan.corr()
plt.figure(figsize=(20,20))
sns.heatmap(cm, annot=True)
plt.show()
```



- This plot shows how closely the features are related.
- It is rated in the form of numbers.
- If the number is closer to 0, then they are not much related to each other.
- If the number is closer to 1, then they are highly related to each other.

Preparing Machine Learning Model :

Identification of task:

This is a regression task as we predict the loan sanction amount.

Preprocessing:

Before training the machine learning model, the data should be preprocessed.

1. Diving the features and target.

   Code:

```
y = loan['Loan Sanction Amount (USD)']
x = loan.drop(['Loan Sanction Amount (USD)'], axis = 1)
```

2. Splitting the dataset into training data and testing data.

   Code:

```
from sklearn.model_selection import train_test_split
x_train,x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

3. Standardizing the data.

   Code:

```
from sklearn.preprocessing import  StandardScaler
standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)
```

Machine Learning Models:

1. Linear Regression Model:

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)    #train the model on training data
linear_predict = lm.predict(x_test)       #predict the output by using testing data and store in variable
linear_accuracy = r2_score(y_test,linear_predict) #measure the accuracy by r2 score
print('r2_score = '+ str(linear_accuracy))
```

r2_score = 0.5669654244336306

2. Decision Tree Regression Model:

```
from sklearn.tree import DecisionTreeRegressor
decisionModel = DecisionTreeRegressor()
decisionModel.fit(x_train,y_train)       #train the model on training data
decisionTree_predict = decisionModel.predict(x_test)      #predict the output by using testing data and store in variable
decisionTree_acuracy = r2_score(y_test, decisionTree_predict)
print('r2_score = '+ str(decisionTree_acuracy))
```

```
r2_score = 0.5065834523078985
```

3. Random Forest Regression Model:

```
from sklearn.ensemble import RandomForestRegressor
#create an instance of the model
forest=RandomForestRegressor()
forest.fit(x_train,y_train)     #train the model on training data
randonForest_predict  =forest.predict(x_test)            #predict  the  output  by
using testing data and store in variable
randomForest_accuracy = r2_score(y_test,randonForest_predict)
print('r2_score = '+ str(randomForest_accuracy))
```

```
r2_score = 0.743050366685013
```

4. Bayesian Ridge Model:

```
from sklearn import linear_model
#create an instance of the model
Bayesian_model = linear_model.BayesianRidge()
Bayesian_model.fit(x_train, y_train)        #train the model on training data
Bayesian_predict = Bayesian_model.predict(x_test)        #predict the output
by using testing data and store in variable
Bayesian_accuracy = r2_score(y_test,Bayesian_predict)
print('r2_score = '+ str(Bayesian_accuracy))
```

```
r2_score = 0.5670822174048955
```

5. Gradient Boosting Regression Model:

```
from sklearn import ensemble
#define params
params = {'n_estimators': 500,
        'max_depth': 4,
        'min_samples_split': 5,
        'learning_rate': 0.01,
        'loss': 'ls'}
#create an instance of the model
gradient_model = ensemble.GradientBoostingRegressor(**params)
gradient_model.fit(x_train, y_train)          #train the model on training data
gradient_predict = gradient_model.predict(x_test)          #predict the output
by using testing data and store in variable
gradient_accuracy = r2_score(y_test, gradient_predict)
print('r2_score = '+ str(gradient_accuracy))
```

```
r2_score = 0.7430255209320876
```

6. XGB Regression Model:

```
import xgboost
xgb = xgboost.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
        colsample_bynode=1, colsample_bytree=1,importance_type='gain', learning_rate=0.1,
        max_depth=5, min_child_weight=1, n_estimators=500, objective='reg:squarederror',
        random_state=42, reg_lambda=1, scale_pos_weight=1,subsample=1, verbosity=1)  #create an instance of the model and pass all the necessary parameters
xgb.fit(x_train, y_train)  #train the model on training data
xgb_predict = xgb.predict(x_test) #predict the output by using testing data and store in variable
xgb_accuracy = r2_score(y_test, xgb_predict)
print('r2_score = '+ str(xgb_accuracy))
```

```
r2_score = 0.7429961073621583
```

7. Lasso regression model:

```
from sklearn import linear_model
lasso = linear_model.Lasso(alpha=1.0, max_iter=1000, tol = 0.0001, random_state = None )
lasso.fit(x_train,y_train)          #train the model on training data
lasso_predict = lasso.predict(x_test)      #predict the output by using testing data and store in variable
lasso_accuracy = r2_score(y_test,lasso_predict)
print('r2_score = '+ str(lasso_accuracy))
```

```
r2_score = 0.5670709061614908
```

8. Ada Boosting model:

```
from sklearn.ensemble import AdaBoostRegressor
adaboost_model = AdaBoostRegressor()      #create an instance of the model
adaboost_model.fit(x_train, y_train)      #train the model on training data
adaboost_predict =adaboost_model.predict(x_test)      #predict the output by using testing data and store in variable
adaboost_accuracy = r2_score(y_test, adaboost_predict)
print('r2_score = '+ str(adaboost_accuracy))
```

```
r2_score = 0.5098268098204687
```

9. K Nearest Neighbours:

```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor()     #create an instance of the model
knn.fit(x_train, y_train)     #train the model on training data
knn_pred = knn.predict(x_test)        #predict the output by using testing data and store in variable
knn_accuracy = r2_score(y_test, knn_pred)
print('r2_score = '+ str(knn_accuracy))
```

```
r2_score = 0.46384960351958426
```

10. Artificial Neural Network:

```
import tensorflow as tf
import warnings
warnings.filterwarnings('ignore')   #ignoring the warnings
ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units = 6, activation='relu'))          #create the
layers
ann.add(tf.keras.layers.Dense(units = 6, activation = 'relu'))
ann.add(tf.keras.layers.Dense(units = 1))
ann.compile(optimizer = 'adam', loss = 'mean_squared_error')
ann.fit(x = x_train,                #train the model on training data by defining
necessary parameters
        y = y_train,
        batch_size = 32,
        epochs = 100,
        shuffle = 1,
        validation_split = 0.05)

annpred = ann.predict(x_test)          #predict the output by using testing data
and store in variable
ann_accuracy = r2_score(y_test,annpred)
print('r2_score = '+ str(ann_accuracy))
```

```
r2_score = 0.5643406305651101
```

ML Model Chart :

Sl. No.	Model Name	R2_score(approx. 4 decimal)
1	Random Forest Regression Model	0.7460
2	XGB Regression Model	0.7429
3	Gradient Boosting Regression Model	0.7429
4	Bayesian Ridge Regression Model	0.5670
5	Lasso Regression Model	0.5670
6	Linear Regression Model	0.5669
7	Artificial Neural Network	0.5662
8	Decision Tree Regression Model	0.4871
9	Ada Boosting Regression Model	0.4754
10	K Near Neighbour Regression Model	0.4638

Hurdles :

- We were worried about the target as it contains negative values also that generally, the loan sanction amount cannot be a negative value.
- We faced a major problem in deciding which should be done first, data cleaning or data visualization, few resources say that data visualization should be done first so that you can analyze the actual data, few others say that data visualization should be done after cleaning data. Few other resources said that it won't be a problem, so we decided to do it in this way.
- Another major problem was researching for the regression models after we applied linear regression, random forest regression and decision tree regression models the r2 score was not satisfying as they were low. We started researching different regression models.

Conclusion :

- By visualizing the data we got to know the relationship between features and also insights from each feature which were helpful to analyse the data.
- Since most of the features seemed to be directly proportional to loan sanction amount, we thought to start by applying a linear regression model and major regression models.
- After applying models we found that the accuracy of each model varied from each other, among them, the best models were Random Forest Regression Model, Gradient Boosting Regression Model, XGB Regression Model.
- But the KNN (K Nearest Neighbours) model came out with low performance compared to all models we applied.

Bibliography :

- Internship Recordings
- Hands-on Machine Learning with scikit-Learn & TensorFlow by Aurelien Geron
- NumPy, Pandas, Scikit-Learn, Seaborn, Matplotlib and TensorFlow.
- https://stackoverflow.com/
- https://www.geeksforgeeks.org/machine-learning/
- https://medium.com/
- YouTube
- Google Scholar
- Machine Learning Tutorial | Machine Learning with Python - Javatpoint