Name: VISHAL HASRAJANI

Email address: vish9221@gmail.com

Contact number: +1-4372445048

Anydesk address: –

Years of Work Experience:

Date: **2022-Sept-27**

Self Case Study -1: Predict the Loan Sanction Amount

Overview

*** Write an overview of the case study that you are working on. ***(MINIMUM 200 words)*** ***

Link for the dataset :https://www.kaggle.com/datasets/zyper26/sanction-loan?select=train.csv

Buying a house requires a lot of careful planning. Once you have finalized your budget and the house that you want to buy, you must ensure that you have sufficient funds to pay the seller.

With rising property rates, most people avail home loans to buy their dream houses. The bank only lends up to 75%- 80% of the total amount based on a person's finances (salary, outgoing expenses, existing loans, etc.). You will need to make the rest of the payment yourself after the bank tells you how much they can lend.

Task:

You work for XYZ bank. Predict the loan amount that can be sanctioned to customers who have applied for a home loan using the features provided in the dataset.

Now to achieve this task, we have to analyze each and every feature and make detailed observations out of it. Also adding new features using domain knowledge would be helpful.After selecting the best features,we can experiment with different machine learning models to predict the best output with minimum error.

My aim is to bring these predictions openly available to the people without even going to the bank.They can just open the website on their device and check if their loan amount can be approved or not Or how much of their loan amount can be approved.

Research-Papers/Solutions/Architectures/Kernels

*** Mention the urls of existing research-papers/solutions/kernels on your problem statement and in your own words write a detailed summary for each one of them. If needed you can include images or explain with your own diagrams. It is mandatory to write a brief description about that paper. Without understanding of the resource please don't mention it***

1. https://www.researchgate.net/publication/360415721_Loan_Prediction_System_Using_Machine_Learning

   Brief Description:

   There are few amazing **feature engineering hacks** that i learnt from this paper that are:

   ➜ **Total Income** : It is a very useful feature.The higher the total income, the more likely there is to get loan approval.

   ➜**EMI** : Monthly installments can also determine that the user can pay back loan or not

   ➜**Balance Income** : The return deserted after compensating the EMI is called Balance Income and this can be a useful feature to determine the profits that the banks are getting from a particular user.

2. https://www.analyticsvidhya.com/blog/2022/02/loan-approval-prediction-machine-learning/#h2_4

This blog also suggests some feature engineering techniques which can be helpful in getting the better accuracy in loan prediction

Some **feature engineering hacks** are as follows:

➔

```
data['Dependents_EMI_mean']=data.groupby(['Dependents'])['EMI'].transform('mean')

# LoanAmount_per_TotalIncome
data['LoanAmount_per_TotalIncome']=data['LoanAmount']/data['TotalIncome']

# Loan_Amount_Term_per_TotalIncome
data['Loan_Amount_Term_per_TotalIncome']=data['Loan_Amount_Term']/data['TotalIncome']

# EMI_per_Loan_Amount_Term
data['EMI_per_Loan_Amount_Term']=data['EMI']/data['Loan_Amount_Term']

# EMI_per_LoanAmount
data['EMI_per_LoanAmount']=data['EMI']/data['LoanAmount']

# Categorical variables wise mean of LoanAmount_per_TotalIncome
data['Property_Area_LoanAmount_per_TotalIncome_mean']=data.groupby(['Property_Area'])['LoanAmount_per_TotalIncome'].transform('mean')

# Credit_History wise sum of TotalIncome
data['Credit_History_Income_Sum']=data.groupby(['Credit_History'])['TotalIncome'].transform('sum')

# Dependents wise sum of LoanAmount
data['Dependents_LoanAmount_Sum']=data.groupby(['Dependents'])['LoanAmount'].transform('sum')
```

BIN Features :

```
from sklearn.preprocessing import KBinsDiscretizer

Loan_Amount_Term_discretizer = KBinsDiscretizer(n_bins=5, encode='ordinal', strategy='quantile')
data['Loan_Amount_Term_Bins'] = Loan_Amount_Term_discretizer.fit_transform(data['Loan_Amount_Term'].values.reshape(-1,1)).astype(float)

TotalIncome_discretizer = KBinsDiscretizer(n_bins=5, encode='ordinal', strategy='quantile')
data['TotalIncome_Bins'] = TotalIncome_discretizer.fit_transform(data['TotalIncome'].values.reshape(-1,1)).astype(float)

LoanAmount_per_TotalIncome_discretizer = KBinsDiscretizer(n_bins=5, encode='ordinal', strategy='quantile')
data['LoanAmount_per_TotalIncome_Bins'] = LoanAmount_per_TotalIncome_discretizer.fit_transform(data['LoanAmount_per_TotalIncome'].values.reshape(-1,1)).astype(float)
```

First Cut Approach

*** Explain in steps about how you want to approach this problem and the initial experiments that you want to do. ***(MINIMUM 200 words)*** ***

*** When you are doing the basic EDA and building the First Cut Approach you should not refer any blogs or papers ***

Ipython notebook for the reference
:https://colab.research.google.com/drive/1qrmVql1I9Nj-ACQeCowjYLZLv8z2OXF5?usp=sharing

Let's start with the dataset first,

➜In this dataset we can observe that there are many **“-999” values in the Target variable** so it's better to **remove those rows** .

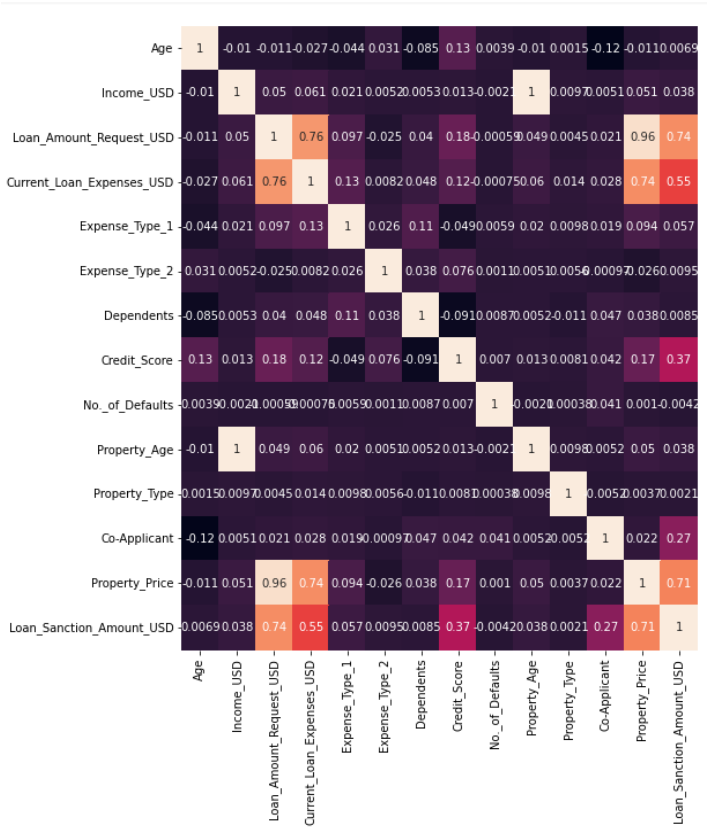
➜Other than this there are **-999 values in some of the columns** ,so we will **replace them will “NaN”**

➜Now we are ready to perform **EDA** .

➜**In EDA**, first we will find if there is any **relation between NaN values and the target variable**(i.e `Loan_Sanction_Amount_USD)`.The relation between them is clearly visible in the **ipython notebook that i have provided above**.So we will replace those NaN values with something meaningful in the later part.

➜Now comes the **analysis on numerical variables** :

➜Here we will start with correlation between the numerical variables :



From the correlation matrix we can note that **"Property Age" and "Income USD"** both have the same values . So **we can drop one of them**.We will drop property Age in the later part**We can also observe that the features named **"Property price " and "Loan amount request USD " have 0.96 collinearity.**

We know that the **bank only lends up to 75%-80% of the total amount** based on an individual's finances (salary, outgoing expenses, existing loans, etc.).**So we will create a **new column which stores the minimum Loan request amount and 75% of property price later.**

➜There are **2 types of numerical variables**:

1)**Discrete** and 2)**Continuous**

1)The most interesting thing while analyzing **discrete variable** was that if **number of Co-Applicant is “0”** then **"Loan Sanction Amount in USD" is also “0”**.Property type doesn't have any value in deciding whether the loan can be sanctioned or not

2) In **continuous variables**, histograms are plotted which can be seen in the ipynb file as mentioned above .The main thing we got here from all these plots is few features have **skewed distribution** and so we can take the **log transformation** in the later analysis.

➜Let's continue our **analysis on categorical variables** :

The only thing that we can observe from the analysis of categorical variables is **if we are "Student" then we have no chance or very less chance to get a loan**.*

Before we start Feature engineering, we can **split our data into train and test.**

Feature Engineering :

➔**Missing values** in **categorical variables** are replaced with the string “**Missing**” .

➔**Missing values** in **numerical variables** are replaced with “**Median Value**” and new features are made where **“1” represents NaN** values and **“0” otherwise**.

➔Replacing the **Category** that is **present in <1%** of the total dataset with “**Rare_var”.**

Additional new features :

1) Minimum of Loan request amount and 75% of property price

2) Interest Rates

E.g.

```
def inrest_rates(row):
    if row['Credit_Score'] <700 :
        val = 13
    elif row['Credit_Score'] >700 and row['Credit_Score'] <750 :
        val = 9
    elif row['Credit_Score'] >750 and row['Credit_Score'] <800 :
        val = 8
    elif row['Credit_Score'] >800 and row['Credit_Score'] <850 :
        val = 7.8
    else:
        val = 7
    return val
```

3) Actual_Term_Suitable=**(**Loan Amount Requested) / (60% of individual's monthly income)

➜ Here 40% can be considered as take home salary and 60% is left from which an individual can pay their loans back.

4) Mean of Loan Amount depending on number of Dependents.

E.g.

```
X_train['Dependent_Loan_Amount_AVG']=X_train.groupby(['Dependents'])['Loan
_Request_Actual'].transform('mean')
```

5) Taking Log Transformation of 3 features as they are skewed

```
['Loan_Request_Actual',
'Current_Loan_Expenses_USD','Actual_Term_Suitable']
```

Finally after converting categorical data into numerical form using One Hot Encoding and also Normalizing data, we observed that 2 newly added features were **adding more value for the final predictions(The features were "Actual_Term_Suitable" and "Interest Rates")**