Name: Fathimah Az Zahra

Tasks: Meeting_1 (Case Study Of Demystifying the Workings of Lending Club)

Point of view: We aim to enhance credit risk management and provide insights into borrower behavior and loan performance.

Case Study:

This project aims to analyze the historical loan data from Lending Club to understand the factors influencing loan defaults and develop predictive models to forecast the likelihood of defaults. By demystifying the workings of Lending Club, we aim to enhance credit risk management and provide insights into borrower behavior and loan performance

The Target:

Lending Club is a peer-to-peer lending platform that connects borrowers with investors. This project focuses on analyzing Lending Club's loan data to predict loan defaults and assess credit risk. By understanding the factors that contribute to loan defaults, we can improve risk management and make data-driven decisions.

Data Preprocessing:

The data preprocessing phase involves the following steps:

1. **Handling Missing Values:** completing or eliminating missing items to guarantee the consistency and completeness of the dataset.
2. **Feature Engineering:** To improve the forecasting ability of the model, extra characteristics like the debt-to-income and payment-to-income ratios can be included.
3. **Encoding and Normalization:** The dataset is prepared for machine learning models by normalizing numerical data and encoding categorical variables.

Exploratory Data Analysis (EDA)

EDA is conducted to understand the distribution of key variables and the relationships between them. Key analyses include:

- Distribution plots for loan amounts, interest rates, and borrower incomes.
- Correlation analysis to identify which factors are most associated with loan defaults.
- Visualization of the default rate across different loan grades and borrower profiles.

Credit Risk Modeling

Several machine learning models are applied to predict loan defaults:

- **Logistic Regression:** A baseline model to understand the impact of different features.
- **Decision Trees:** To capture non-linear relationships between features and the default status.
- **Random Forest:** An ensemble method to improve predictive accuracy and reduce overfitting.
- **Gradient Boosting:** To optimize predictive performance by sequentially minimizing errors.

Loan Default Prediction

The prediction phase involves:

- Implementing algorithms to predict the likelihood of loan defaults.
- Optimizing decision thresholds to balance sensitivity (true positive rate) and specificity (true negative rate).
- Assessing model performance using key metrics and refining the models as needed.

Some processes that can be done based on two different data:

1. **lending_club_loans dataset**

   The dataset serves as a Lending Club data dictionary, with thorough justifications for all of the variables that are used. It acts as a guide to assist comprehend the significance and context of each variable, which facilitates accurate data analysis and interpretation. The remaining columns don't add any new information and are mostly blank.

2. **Lcdatadictionary dataset**

   The Lending Club data dictionary offers thorough explanations of each variable included in the dataset. To aid users in correctly analyzing and comprehending the data, each row in the dataset contains the name of the variable along with an explanation of what the variable represents. In order to ensure accurate variable interpretation throughout additional analysis, this data dictionary is crucial.

Preprocessing data from two excel using Python:

1. **Lcdatadictionary dataset**
   - **Excell View:**

LoanStatNew	Description
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
acc_open_past_24mths	Number of trades opened in past 24 months.
addr_state	The state provided by the borrower in the loan application
all_util	Balance to credit limit on all trades
annual_inc	The self-reported annual income provided by the borrower during registration.
annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
application_type	Indicates whether the loan is an individual application or a joint application with two co-
avg_cur_bal	Average current balance of all accounts
bc_open_to_buy	Total open to buy on revolving bankcards.
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
chargeoff_within_12_mths	Number of charge-offs within 12 months
collection_recovery_fee	post charge off collection fee
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
desc	Loan description provided by the borrower
dti	A ratio calculated using the borrower’s total monthly debt payments on the total debt oblig
dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligat
earliest_cr_line	The month the borrower's earliest reported credit line was opened

Note: Show the dataset.

- **Python View:**

a. Show the dataset:

```
data_dict = pd.read_csv("LCDataDictionary.csv", delimiter=';', low_memory=False)
data_dict.head()
```

[13] ✓ 0.0s

	LoanStatNew,Description
0	acc_now_delinq,The number of accounts on which...
1	acc_open_past_24mths,Number of trades opened i...
2	addr_state,The state provided by the borrower ...
3	all_util,Balance to credit limit on all trades
4	annual_inc,The self-reported annual income pro...

Note: Show the dataset values.

b. Drop Missing Value:

```
missing_values = data_dict.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

[14] ✓ 0.0s

```
Missing values in each column:
 LoanStatNew,Description    0
dtype: int64
```

Note: The total number of missing values for each column is represented by each entry in the Series that this variable holds. This data is then output by the print command, giving a brief summary of the number of missing values in each dataset column.

c. Removing Duplicates:

```
# Remove duplicate rows
dict_data_cleaned = dict_data_cleaned.drop_duplicates()

# Verify that there are no duplicates
print("Number of duplicates after cleaning: ", dict_data_cleaned.duplicated().sum())
```

[17] ✓ 0.0s

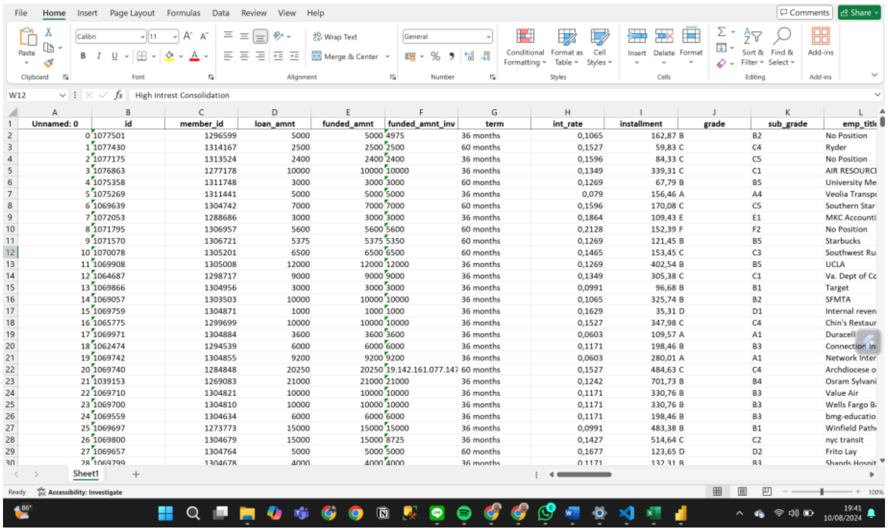
```
Number of duplicates after cleaning:  0
```

Note: Check the duplicates value after cleaning.

2. lending_club_loans dataset

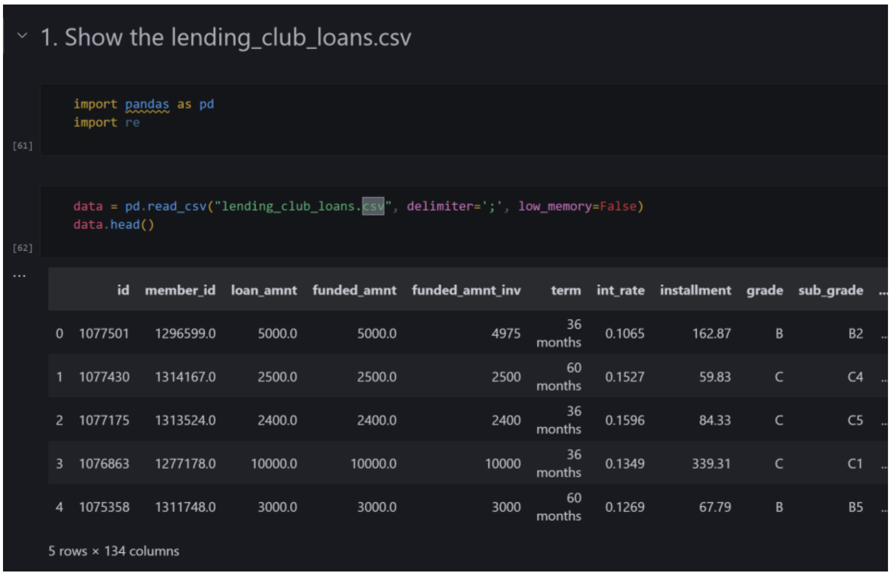
- **Excel View:**

  After cleaning and removing duplicate values. For some empty columns, temporarily delete them so that they do not become empty columns.



- **Python View:**

  a. Show the dataset:



1. Show the lending_club_loans.csv

```python
import pandas as pd
import re
```

[61]

```python
data = pd.read_csv("lending_club_loans.csv", delimiter=';', low_memory=False)
data.head()
```

[62]

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...
0	1077501	1296599.0	5000.0	5000.0	4975	36 months	0.1065	162.87	B	B2	...
1	1077430	1314167.0	2500.0	2500.0	2500	60 months	0.1527	59.83	C	C4	...
2	1077175	1313524.0	2400.0	2400.0	2400	36 months	0.1596	84.33	C	C5	...
3	1076863	1277178.0	10000.0	10000.0	10000	36 months	0.1349	339.31	C	C1	...
4	1075358	1311748.0	3000.0	3000.0	3000	60 months	0.1269	67.79	B	B5	...

5 rows × 134 columns

Note: Show the dataset values.

b. Drop Missing Value:

2. DROP Missing Value (lending_club_loans.csv)

```
kolom_dihapus = ['mths_since_last_major_derog', 'annual_inc_joint', 'dti_joint', 'verification_status_joint', 'tot_coll_amt', 'tot_cur_bal', 'open_acc_
                 'max_bal_bc', 'all_util', 'total_rev_hi_lim', 'inq_fi', 'total_cu_tl', 'inq_last_12m', 'acc_open_past_24mths', 'avg_cur_bal', 'bc_open
                 'mths_since_recent_revol_delinq', 'num_accts_ever_120_pd', 'num_actv_bc_tl', 'num_actv_rev_tl', 'num_bc_sats', 'num_bc_tl', 'num_il_tl
                 'pct_tl_nvr_dlq', 'percent_bc_gt_75', 'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit', 'total_il_high_credit_limit', 'Unnamed
                 'Unnamed: 120','Unnamed: 121','Unnamed: 122','Unnamed: 123','Unnamed: 124','Unnamed: 125','Unnamed: 126','Unnamed: 127','Unnamed: 128'
data = data.drop(kolom_dihapus, axis=1)
```

[64]

```
data.info()
```

[65]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42536 entries, 0 to 42535
Data columns (total 61 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    42536 non-null  object
 1   member_id             42535 non-null  float64
 2   loan_amnt             42535 non-null  float64
 3   funded_amnt           42535 non-null  float64
 4   funded_amnt_inv       42535 non-null  object
 5   term                  42535 non-null  object
 6   int_rate              42535 non-null  float64
 7   installment           42535 non-null  float64
 8   grade                 42535 non-null  object
 9   sub_grade             42535 non-null  object
 10  emp_title             42528 non-null  object
 11  emp_length            41417 non-null  object
 12  home_ownership        42536 non-null  object
```

Note: This code uses a list of column names kept in the variable {kolom_dihapus} to delete certain columns from the DataFrame {data}. The list includes names of columns that ought to be removed from the dataset; these include ones that might be superfluous or useless, like {Unnamed} or other columns with ambiguous names. All of the columns indicated in {kolom_dihapus} are deleted from the DataFrame {data} by running `data.drop(kolom_dihapus, axis=1)}, leaving only the desired columns in the dataset.

```
data.isnull().sum().reset_index()
```

[66]

	index	0
0	id	0
1	member_id	1
2	loan_amnt	1
3	funded_amnt	1
4	funded_amnt_inv	1
...	...	...
56	acc_now_delinq	1736
57	chargeoff_within_12_mths	1856
58	delinq_amnt	1739
59	pub_rec_bankruptcies	3055
60	tax_liens	1815

61 rows × 2 columns

Note: With each row representing a column in the original data DataFrame and displaying the column name and the amount of missing values it contains, the code generates a DataFrame. This helps to rapidly determine which columns and to what extent are lacking data.

```
[71]  data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42536 entries, 0 to 42535
Data columns (total 61 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      42536 non-null  object
 1   member_id               42535 non-null  float64
 2   loan_amnt               42535 non-null  float64
 3   funded_amnt             42535 non-null  float64
 4   funded_amnt_inv         42535 non-null  object
 5   term                    42535 non-null  object
 6   int_rate                42535 non-null  float64
 7   installment             42535 non-null  float64
 8   grade                   42535 non-null  object
 9   sub_grade               42535 non-null  object
 10  emp_title               42528 non-null  object
 11  emp_length              41417 non-null  object
 12  home_ownership          42536 non-null  object
 13  annual_inc              42536 non-null  float64
 14  verification_status     42529 non-null  object
 15  issue_d                 42529 non-null  object
 16  loan_status             42529 non-null  object
 17  pymnt_plan              42529 non-null  object
 18  url                     42529 non-null  object
 19  desc                    29234 non-null  object
...
 59  pub_rec_bankruptcies    39481 non-null  float64
 60  tax_liens               40721 non-null  object
dtypes: float64(8), object(53)
memory usage: 19.8+ MB
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Note: The data.info() function gives a summary of the DataFrame, including its memory utilization, number of rows and columns, data types for each column, and number of non-null entries.

c. Removing Duplicates:

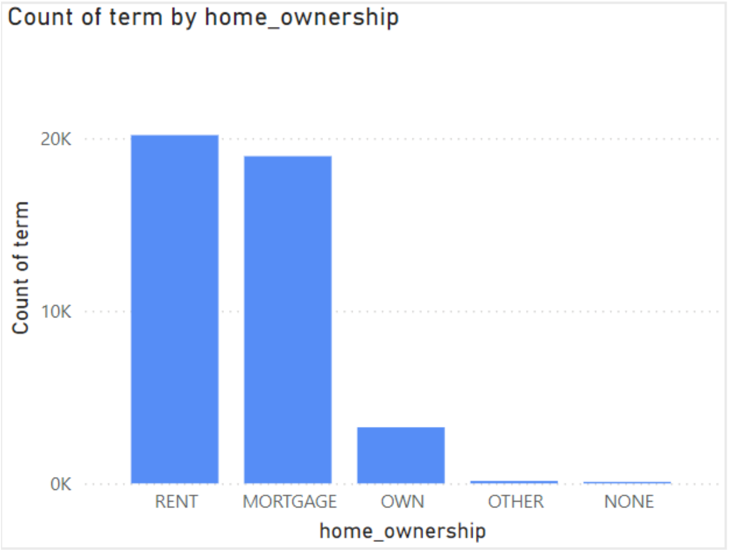
```
[67]  data_bersih_kolom_seluruhnya_nan = data.dropna(axis=1, how='all')

[68]  data.shape

(42536, 61)
```

Note: The dataset that remains contains just columns with some valid data after these fully-NaN columns are eliminated and are then saved in the variable data_bersih_kolom_seluruhnya_nan.

3. Visualization using Power BI

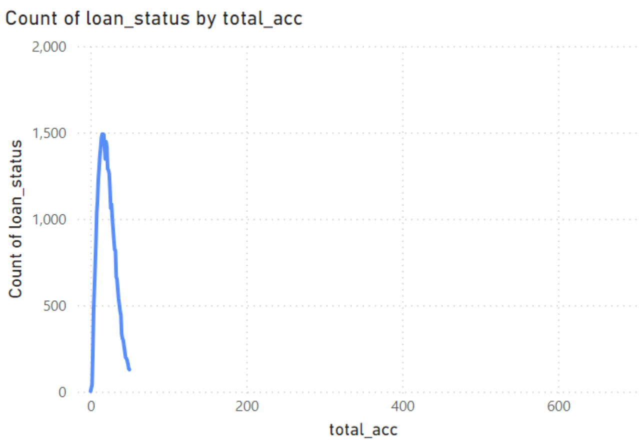
Due to the large number of columns dropped in excell lending_club_loans, there are not many visualizations that can be created. However, these columns can be created using calculations in Power BI using existing columns to get new values according to what is given in the description of each empty column.

1. **Count of Term by Home Ownership**



Note: The bar chart displays the frequency with which each category of house ownership occurs in the dataset relative to the length of the loan. Given that these are the most prevalent house ownership statuses among the borrowers in this sample, RENT and MORTGAGE have the greatest counts.

2. **Count of Loan Status by Total Accounts**



Note: The distribution of the total number of accounts (total_acc) in relation to the counts of loans in status is displayed on the line graph. With a sharp decline as the overall number of accounts rises, the peak shows the areas where the majority of loan statuses are concentrated in respect to the number of accounts, indicating that the majority of borrowers have comparatively few accounts.

Previously I apologize if I have not done my best in completing the assignment, I will continue to try to do future assignments better. Thank you for giving me this task I am happy to be able to contribute.