Loan_Default_Risk_final

September 20, 2024

1 Hackathon Challenge: Predicting Loan Default Risk.

Financial institutions face significant challenges in assessing the creditworthiness of loan applicants. Accurate evaluation of an applicant's ability to repay a loan is crucial for minimising defaults and managing risk. Traditional methods of credit assessment may not fully capture the complexities of individual financial behaviour, leading to potential losses for lenders. By leveraging machine learning and predictive analytics, institutions can enhance their loan approval processes and make more informed decisions.

1.1 Challenge:

Develop a predictive model that accurately evaluates the risk of loan defaults based on applicant data. The model should assist financial institutions in making better lending decisions, ultimately reducing the risk of defaults and improving overall portfolio performance.

1.2 Objectives:

`1). Data Exploration and Preprocessing:`

Analyse the dataset to identify key features that influence loan default risk. Perform necessary data cleaning and preprocessing to ensure the dataset is ready for modelling.

`2). Model Development:`

Build and train a predictive model to evaluate the likelihood of loan defaults. Explore and implement various machine learning algorithms such as Logistic Regression, Decision Trees, Random Forest, Gradient Boosting Machines, etc.

`3). Evaluation:`

Assess the model's performance using relevant metrics such as accuracy, precision, recall, F1 score, ROC-AUC, and especially the precision-recall trade-off to balance risk and reward. Ensure the model's predictions are fair and unbiased, particularly in sensitive areas like income and employment status.

`4). Actionable Insights:`

Provide insights into the factors that most significantly impact loan default risk. Offer recommendations on how financial institutions can integrate the model into their loan approval processes to enhance decision-making and risk management.

`5) . Documentation and Presentation:`

Document your process, including data preprocessing, model selection, and evaluation criteria. Prepare a presentation that clearly communicates the model's performance, insights, and practical implications for financial institutions . ## Submission Requirements: A well-documented code repository with instructions for running your model. A detailed report or presentation summarising your approach, results, and insights. A demo or visualisation showcasing the model's predictions and its potential impact on loan approval processes.

1.3 Evaluation Criteria:

Accuracy and performance of the predictive model. Quality and thoroughness of data preprocessing and feature engineering. Fairness and lack of bias in predictions. Clarity and usefulness of actionable insights provided. Overall presentation and documentation.

1.4 Dataset courtesy of kaggle https://www.kaggle.com/datasets/yasserh/loan-default-dataset

[ ]:
```
%pip install pycaret
```

```
Collecting pycaret
  Downloading pycaret-3.3.2-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: ipython>=5.5.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (7.34.0)
Requirement already satisfied: ipywidgets>=7.6.5 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (7.7.1)
Requirement already satisfied: tqdm>=4.62.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (4.66.5)
Requirement already satisfied: numpy<1.27,>=1.21 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (1.26.4)
Requirement already satisfied: pandas<2.2.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (2.1.4)
Requirement already satisfied: jinja2>=3 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (3.1.4)
Collecting scipy<=1.11.4,>=1.6.1 (from pycaret)
  Downloading
scipy-1.11.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(60 kB)
                           60.4/60.4 kB
987.1 kB/s eta 0:00:00
Collecting joblib<1.4,>=1.2.0 (from pycaret)
  Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
Collecting scikit-learn>1.4.0 (from pycaret)
  Downloading scikit_learn-1.5.2-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Collecting pyod>=1.1.3 (from pycaret)
  Downloading pyod-2.0.2.tar.gz (165 kB)
                           165.8/165.8

kB 2.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
```

```
Requirement already satisfied: imbalanced-learn>=0.12.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (0.12.3)
Collecting category-encoders>=2.4.0 (from pycaret)
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl.metadata (8.0 kB)
Requirement already satisfied: lightgbm>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (4.5.0)
Requirement already satisfied: numba>=0.55.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (0.60.0)
Requirement already satisfied: requests>=2.27.1 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (2.32.3)
Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (5.9.5)
Requirement already satisfied: markupsafe>=2.0.1 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (2.1.5)
Requirement already satisfied: importlib-metadata>=4.12.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (8.5.0)
Requirement already satisfied: nbformat>=4.2.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (5.10.4)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-
packages (from pycaret) (2.2.1)
Collecting deprecation>=2.1.0 (from pycaret)
  Downloading deprecation-2.1.0-py2.py3-none-any.whl.metadata (4.6 kB)
Collecting xxhash (from pycaret)
  Downloading
xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(12 kB)
Requirement already satisfied: matplotlib<3.8.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (3.7.1)
Collecting scikit-plot>=0.3.7 (from pycaret)
  Downloading scikit_plot-0.3.7-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: yellowbrick>=1.4 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (1.5)
Requirement already satisfied: plotly>=5.14.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (5.15.0)
Collecting kaleido>=0.2.1 (from pycaret)
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
Collecting schemdraw==0.15 (from pycaret)
  Downloading schemdraw-0.15-py3-none-any.whl.metadata (2.2 kB)
Collecting plotly-resampler>=0.8.3.1 (from pycaret)
  Downloading plotly_resampler-0.10.0-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: statsmodels>=0.12.1 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (0.14.3)
Collecting sktime==0.26.0 (from pycaret)
  Downloading sktime-0.26.0-py3-none-any.whl.metadata (29 kB)
Collecting tbats>=1.1.3 (from pycaret)
  Downloading tbats-1.1.3-py3-none-any.whl.metadata (3.8 kB)
Collecting pmdarima>=2.0.4 (from pycaret)
  Downloading pmdarima-2.0.4-cp310-cp310-
```

```
manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata
(7.8 kB)
Collecting wurlitzer (from pycaret)
  Downloading wurlitzer-3.1.1-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from sktime==0.26.0->pycaret) (24.1)
Collecting scikit-base<0.8.0 (from sktime==0.26.0->pycaret)
  Downloading scikit_base-0.7.8-py3-none-any.whl.metadata (8.8 kB)
Collecting scikit-learn>1.4.0 (from pycaret)
  Downloading scikit_learn-1.4.2-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-
packages (from category-encoders>=2.4.0->pycaret) (0.5.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn>=0.12.0->pycaret)
(3.5.0)
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.10/dist-
packages (from importlib-metadata>=4.12.0->pycaret) (3.20.2)
Requirement already satisfied: setuptools>=18.5 in
/usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (71.0.4)
Collecting jedi>=0.16 (from ipython>=5.5.0->pycaret)
  Using cached jedi-0.19.1-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (5.7.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (3.0.47)
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (2.18.0)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (0.2.0)
Requirement already satisfied: matplotlib-inline in
/usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (4.9.0)
Requirement already satisfied: ipykernel>=4.5.1 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(5.5.6)
Requirement already satisfied: ipython-genutils~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(0.2.0)
Requirement already satisfied: widgetsnbextension~=3.6.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(3.6.9)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(3.0.13)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<3.8.0->pycaret) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret)
(4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (1.4.7)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<3.8.0->pycaret) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<3.8.0->pycaret) (2.8.2)
Requirement already satisfied: fastjsonschema>=2.15 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=4.2.0->pycaret) (2.20.0)
Requirement already satisfied: jsonschema>=2.6 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=4.2.0->pycaret) (4.23.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=4.2.0->pycaret) (5.7.2)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in
/usr/local/lib/python3.10/dist-packages (from numba>=0.55.0->pycaret) (0.43.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<2.2.0->pycaret) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<2.2.0->pycaret) (2024.1)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly>=5.14.0->pycaret) (9.0.0)
Collecting dash>=2.9.0 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading dash-2.18.1-py3-none-any.whl.metadata (10 kB)
Collecting orjson<4.0.0,>=3.8.0 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading orjson-3.10.7-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (50 kB)
                            50.4/50.4 kB
3.0 MB/s eta 0:00:00
Collecting tsdownsample>=0.1.3 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading tsdownsample-0.1.3-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (7.9 kB)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in
/usr/local/lib/python3.10/dist-packages (from pmdarima>=2.0.4->pycaret) (3.0.11)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-
packages (from pmdarima>=2.0.4->pycaret) (2.0.7)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.27.1->pycaret) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
```

```
packages (from requests>=2.27.1->pycaret) (3.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.27.1->pycaret)
(2024.8.30)
Requirement already satisfied: Flask<3.1,>=1.0.4 in
/usr/local/lib/python3.10/dist-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (2.2.5)
Requirement already satisfied: Werkzeug<3.1 in /usr/local/lib/python3.10/dist-
packages (from dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret) (3.0.4)
Collecting dash-html-components==2.0.0 (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading dash_html_components-2.0.0-py3-none-any.whl.metadata (3.8 kB)
Collecting dash-core-components==2.0.0 (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading dash_core_components-2.0.0-py3-none-any.whl.metadata (2.9 kB)
Collecting dash-table==5.0.0 (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading dash_table-5.0.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: typing-extensions>=4.1.1 in
/usr/local/lib/python3.10/dist-packages (from dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (4.12.2)
Collecting retrying (from dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret)
  Downloading retrying-1.3.4-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.10/dist-
packages (from dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret) (1.6.0)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.10/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.6.5->pycaret) (6.1.12)
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.10/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.6.5->pycaret) (6.3.3)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from
jedi>=0.16->ipython>=5.5.0->pycaret) (0.8.4)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=4.2.0->pycaret) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=4.2.0->pycaret) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=4.2.0->pycaret) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=4.2.0->pycaret) (0.20.0)
Requirement already satisfied: platformdirs>=2.5 in
/usr/local/lib/python3.10/dist-packages (from jupyter-
core!=5.0.*,>=4.12->nbformat>=4.2.0->pycaret) (4.3.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from patsy>=0.5.1->category-encoders>=2.4.0->pycaret) (1.16.0)
Requirement already satisfied: ptyprocess>=0.5 in
```

```
/usr/local/lib/python3.10/dist-packages (from
pexpect>4.3->ipython>=5.5.0->pycaret) (0.7.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-
packages (from prompt-
toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=5.5.0->pycaret) (0.2.13)
Requirement already satisfied: notebook>=4.4.1 in
/usr/local/lib/python3.10/dist-packages (from
widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (6.5.5)
Requirement already satisfied: itsdangerous>=2.0 in
/usr/local/lib/python3.10/dist-packages (from
Flask<3.1,>=1.0.4->dash>=2.9.0->plotly-resampler>=0.8.3.1->pycaret) (2.2.0)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-
packages (from Flask<3.1,>=1.0.4->dash>=2.9.0->plotly-
resampler>=0.8.3.1->pycaret) (8.1.7)
Requirement already satisfied: pyzmq<25,>=17 in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (24.0.1)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (23.1.0)
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (6.5.4)
Requirement already satisfied: Send2Trash>=1.8.0 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.8.3)
Requirement already satisfied: terminado>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.18.1)
Requirement already satisfied: prometheus-client in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.20.0)
Requirement already satisfied: nbclassic>=0.4.7 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.1.0)
Requirement already satisfied: notebook-shim>=0.2.3 in
/usr/local/lib/python3.10/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.2.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
```

```
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.4)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.3.0)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.10.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.5.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (1.3.0)
Requirement already satisfied: argon2-cffi-bindings in
/usr/local/lib/python3.10/dist-packages (from argon2-cffi->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (21.2.0)
Requirement already satisfied: jupyter-server<3,>=1.8 in
/usr/local/lib/python3.10/dist-packages (from notebook-shim>=0.2.3-
>nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (1.24.0)
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.17.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (2.6)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach->nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (0.5.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-
packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (2.22)
Requirement already satisfied: anyio<4,>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from jupyter-server<3,>=1.8->notebook-s
him>=0.2.3->nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (3.7.1)
Requirement already satisfied: websocket-client in
/usr/local/lib/python3.10/dist-packages (from jupyter-server<3,>=1.8->notebook-s
him>=0.2.3->nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (1.8.0)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-
```

```
packages (from anyio<4,>=3.1.0->jupyter-server<3,>=1.8->notebook-shim>=0.2.3-
>nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio<4,>=3.1.0->jupyter-server<3,>=1.8->notebook-shim>=0.2.3-
>nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (1.2.2)
Downloading pycaret-3.3.2-py3-none-any.whl (486 kB)
                         486.1/486.1 kB
8.8 MB/s eta 0:00:00
Downloading schemdraw-0.15-py3-none-any.whl (106 kB)
                         106.8/106.8 kB
7.7 MB/s eta 0:00:00
Downloading sktime-0.26.0-py3-none-any.whl (21.8 MB)
                         21.8/21.8 MB
42.7 MB/s eta 0:00:00
Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
                         81.9/81.9 kB
4.7 MB/s eta 0:00:00
Downloading deprecation-2.1.0-py2.py3-none-any.whl (11 kB)
Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
                         302.2/302.2 kB
18.7 MB/s eta 0:00:00
Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
                         79.9/79.9 MB
6.9 MB/s eta 0:00:00
Downloading plotly_resampler-0.10.0-py3-none-any.whl (80 kB)
                         80.7/80.7 kB
4.8 MB/s eta 0:00:00
Downloading pmdarima-2.0.4-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
                         2.1/2.1 MB
72.5 MB/s eta 0:00:00
Downloading
scikit_learn-1.4.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(12.1 MB)
                         12.1/12.1 MB
67.9 MB/s eta 0:00:00
Downloading scikit_plot-0.3.7-py3-none-any.whl (33 kB)
Downloading
scipy-1.11.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (36.4
MB)
                         36.4/36.4 MB
16.4 MB/s eta 0:00:00
Downloading tbats-1.1.3-py3-none-any.whl (44 kB)
                         44.0/44.0 kB
3.0 MB/s eta 0:00:00
Downloading wurlitzer-3.1.1-py3-none-any.whl (8.6 kB)
```

```
Downloading
xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
                         194.1/194.1 kB
14.2 MB/s eta 0:00:00
Downloading dash-2.18.1-py3-none-any.whl (7.5 MB)
                         7.5/7.5 MB
69.4 MB/s eta 0:00:00
Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Using cached jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
Downloading
orjson-3.10.7-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (141
kB)
                         141.9/141.9 kB
10.2 MB/s eta 0:00:00
Downloading scikit_base-0.7.8-py3-none-any.whl (130 kB)
                         130.1/130.1 kB
10.6 MB/s eta 0:00:00
Downloading
tsdownsample-0.1.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(2.1 MB)
                         2.1/2.1 MB
57.0 MB/s eta 0:00:00
Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Building wheels for collected packages: pyod
  Building wheel for pyod (setup.py) … done
  Created wheel for pyod: filename=pyod-2.0.2-py3-none-any.whl size=198469
sha256=4d38a2acdd8668800eebf750e6d5baea1aa408d1c8d4131beebc0cdda812b72e
  Stored in directory: /root/.cache/pip/wheels/77/c2/20/34d1f15b41b701ba69f42a32
304825810d680754d509f91391
Successfully built pyod
Installing collected packages: kaleido, dash-table, dash-html-components, dash-
core-components, xxhash, wurlitzer, tsdownsample, scipy, scikit-base, schemdraw,
retrying, orjson, joblib, jedi, deprecation, scikit-learn, sktime, scikit-plot,
pyod, dash, pmdarima, plotly-resampler, category-encoders, tbats, pycaret
  Attempting uninstall: scipy
    Found existing installation: scipy 1.13.1
    Uninstalling scipy-1.13.1:
      Successfully uninstalled scipy-1.13.1
  Attempting uninstall: joblib
    Found existing installation: joblib 1.4.2
    Uninstalling joblib-1.4.2:
      Successfully uninstalled joblib-1.4.2
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.3.2
    Uninstalling scikit-learn-1.3.2:
      Successfully uninstalled scikit-learn-1.3.2
```

```
Successfully installed category-encoders-2.6.3 dash-2.18.1 dash-core-
components-2.0.0 dash-html-components-2.0.0 dash-table-5.0.0 deprecation-2.1.0
jedi-0.19.1 joblib-1.3.2 kaleido-0.2.1 orjson-3.10.7 plotly-resampler-0.10.0
pmdarima-2.0.4 pycaret-3.3.2 pyod-2.0.2 retrying-1.3.4 schemdraw-0.15 scikit-
base-0.7.8 scikit-learn-1.4.2 scikit-plot-0.3.7 scipy-1.11.4 sktime-0.26.0
tbats-1.1.3 tsdownsample-0.1.3 wurlitzer-3.1.1 xxhash-3.5.0
```

[ ]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.impute import KNNImputer, SimpleImputer  # to be used for NaN␣
 ↪imputation

from pycaret.classification import *
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

#Resampling
from imblearn.over_sampling import SMOTE

# models
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,␣
 ↪GradientBoostingClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
import xgboost as xgb
import lightgbm as lgb
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from lightgbm import LGBMClassifier
```

1.4.1 Import dataset

[ ]:
```python
df = pd.read_csv('/content/Loan_Default.csv')
df.head()
```

[ ]:
```
      ID  year loan_limit             Gender approv_in_adv loan_type  \
0  24890  2019         cf  Sex Not Available         nopre     type1
1  24891  2019         cf               Male         nopre     type2
```

```
2  24892  2019          cf               Male           pre     type1
3  24893  2019          cf               Male         nopre     type1
4  24894  2019          cf              Joint           pre     type1

  loan_purpose Credit_Worthiness open_credit business_or_commercial  …  \
0           p1                l1        nopc                  nob/c  …
1           p1                l1        nopc                    b/c  …
2           p1                l1        nopc                  nob/c  …
3           p4                l1        nopc                  nob/c  …
4           p1                l1        nopc                  nob/c  …

   credit_type  Credit_Score  co-applicant_credit_type    age  \
0          EXP           758                       CIB  25-34
1         EQUI           552                       EXP  55-64
2          EXP           834                       CIB  35-44
3          EXP           587                       CIB  45-54
4         CRIF           602                       EXP  25-34

   submission_of_application        LTV Region Security_Type  Status dtir1
0                    to_inst  98.728814  south        direct       1  45.0
1                    to_inst        NaN  North        direct       1   NaN
2                    to_inst  80.019685  south        direct       0  46.0
3                   not_inst  69.376900  North        direct       0  42.0
4                   not_inst  91.886544  North        direct       0  39.0

[5 rows x 34 columns]
```

[ ]:
```
df.tail()
```

[ ]:
```
            ID  year loan_limit             Gender approv_in_adv loan_type  \
148665  173555  2019         cf  Sex Not Available         nopre     type1
148666  173556  2019         cf               Male         nopre     type1
148667  173557  2019         cf               Male         nopre     type1
148668  173558  2019         cf             Female         nopre     type1
148669  173559  2019         cf             Female         nopre     type1

       loan_purpose Credit_Worthiness open_credit business_or_commercial  …  \
148665           p3                l1        nopc                  nob/c  …
148666           p1                l1        nopc                  nob/c  …
148667           p4                l1        nopc                  nob/c  …
148668           p4                l1        nopc                  nob/c  …
148669           p3                l1        nopc                  nob/c  …

        credit_type  Credit_Score  co-applicant_credit_type    age  \
148665          CIB           659                       EXP  55-64
148666          CIB           569                       CIB  25-34
148667          CIB           702                       EXP  45-54
```

```
148668          EXP          737                        EXP  55-64
148669          CIB          830                        CIB  45-54

        submission_of_application        LTV Region Security_Type  Status  \
148665                    to_inst  71.792763  south        direct       0
148666                   not_inst  74.428934  south        direct       0
148667                   not_inst  61.332418  North        direct       0
148668                    to_inst  70.683453  North        direct       0
148669                   not_inst  72.849462  North        direct       0

       dtir1
148665  48.0
148666  15.0
148667  49.0
148668  29.0
148669  44.0

[5 rows x 34 columns]
```

[ ]:
```
#Check the ddata types,
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148670 entries, 0 to 148669
Data columns (total 34 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   ID                         148670 non-null  int64
 1   year                       148670 non-null  int64
 2   loan_limit                 145326 non-null  object
 3   Gender                     148670 non-null  object
 4   approv_in_adv              147762 non-null  object
 5   loan_type                  148670 non-null  object
 6   loan_purpose               148536 non-null  object
 7   Credit_Worthiness          148670 non-null  object
 8   open_credit                148670 non-null  object
 9   business_or_commercial     148670 non-null  object
 10  loan_amount                148670 non-null  int64
 11  rate_of_interest           112231 non-null  float64
 12  Interest_rate_spread       112031 non-null  float64
 13  Upfront_charges            109028 non-null  float64
 14  term                       148629 non-null  float64
 15  Neg_ammortization          148549 non-null  object
 16  interest_only              148670 non-null  object
 17  lump_sum_payment           148670 non-null  object
 18  property_value             133572 non-null  float64
 19  construction_type          148670 non-null  object
 20  occupancy_type             148670 non-null  object
```

```
 21  Secured_by                 148670 non-null  object
 22  total_units                148670 non-null  object
 23  income                     139520 non-null  float64
 24  credit_type                148670 non-null  object
 25  Credit_Score               148670 non-null  int64
 26  co-applicant_credit_type   148670 non-null  object
 27  age                        148470 non-null  object
 28  submission_of_application  148470 non-null  object
 29  LTV                        133572 non-null  float64
 30  Region                     148670 non-null  object
 31  Security_Type              148670 non-null  object
 32  Status                     148670 non-null  int64
 33  dtir1                      124549 non-null  float64
dtypes: float64(8), int64(5), object(21)
memory usage: 38.6+ MB
```

[ ]:
```python
# Check number of rows and columns in the data frame
df.shape
```

[ ]:
```
(148670, 34)
```

[ ]:
```python
#Check the df data traits for numeric columns ((ignore ID and year columns))
df.describe().T
```

[ ]:
```
                        count           mean            std           min  \
ID                   148670.0   99224.500000   42917.476598  24890.000000
year                 148670.0    2019.000000       0.000000   2019.000000
loan_amount          148670.0  331117.743997  183909.310127  16500.000000
rate_of_interest     112231.0       4.045476       0.561391      0.000000
Interest_rate_spread 112031.0       0.441656       0.513043     -3.638000
Upfront_charges      109028.0    3224.996127    3251.121510      0.000000
term                 148629.0     335.136582      58.409084     96.000000
property_value       133572.0  497893.465696  359935.315562   8000.000000
income               139520.0    6957.338876    6496.586382      0.000000
Credit_Score         148670.0     699.789103     115.875857    500.000000
LTV                  133572.0      72.746457      39.967603      0.967478
Status               148670.0       0.246445       0.430942      0.000000
dtir1                124549.0      37.732932      10.545435      5.000000

                            25%           50%            75%           max
ID                  62057.25000   99224.50000  136391.750000  1.735590e+05
year                 2019.00000    2019.00000    2019.000000  2.019000e+03
loan_amount        196500.00000  296500.00000  436500.000000  3.576500e+06
rate_of_interest        3.62500       3.99000       4.375000  8.000000e+00
Interest_rate_spread    0.07600       0.39040       0.775400  3.357000e+00
Upfront_charges       581.49000    2596.45000    4812.500000  6.000000e+04
term                  360.00000     360.00000     360.000000  3.600000e+02
property_value     268000.00000  418000.00000  628000.000000  1.650800e+07
```

```
income                    3720.00000    5760.00000    8520.000000  5.785800e+05
Credit_Score               599.00000     699.00000     800.000000  9.000000e+02
LTV                         60.47486      75.13587      86.184211  7.831250e+03
Status                       0.00000       0.00000       0.000000  1.000000e+00
dtir1                       31.00000      39.00000      45.000000  6.100000e+01
```

[ ]:
```
#Check the total missing values for each column
df.isna().sum()
```

[ ]:
```
ID                               0
year                             0
loan_limit                    3344
Gender                           0
approv_in_adv                  908
loan_type                        0
loan_purpose                   134
Credit_Worthiness                0
open_credit                      0
business_or_commercial           0
loan_amount                      0
rate_of_interest             36439
Interest_rate_spread         36639
Upfront_charges              39642
term                            41
Neg_ammortization              121
interest_only                    0
lump_sum_payment                 0
property_value               15098
construction_type                0
occupancy_type                   0
Secured_by                       0
total_units                      0
income                        9150
credit_type                      0
Credit_Score                     0
co-applicant_credit_type         0
age                            200
submission_of_application      200
LTV                          15098
Region                           0
Security_Type                    0
Status                           0
dtir1                        24121
dtype: int64
```

2 Step (II): Data Cleaning

Handling Missing Values

2.0.1 Check Information on the different columns (to identify categorical and numeric columns)

[ ]:
```
columns = df.columns
```

[ ]:
```
#This will help in identifying categorical columns that are in numeric form
for column in columns:
  print(f"\n\033[1m{column}\033[0m\n")  #  prints column name in bold
  print(df[column].unique()[:10]) # print the first 10
```

ID

```
[24890 24891 24892 24893 24894 24895 24896 24897 24898 24899]
```

year

```
[2019]
```

loan_limit

```
['cf' nan 'ncf']
```

Gender

```
['Sex Not Available' 'Male' 'Joint' 'Female']
```

approv_in_adv

```
['nopre' 'pre' nan]
```

loan_type

```
['type1' 'type2' 'type3']
```

loan_purpose

```
['p1' 'p4' 'p3' 'p2' nan]
```

Credit_Worthiness

```
['l1' 'l2']
```

open_credit

['nopc' 'opc']

business_or_commercial

['nob/c' 'b/c']

loan_amount

[116500 206500 406500 456500 696500 706500 346500 266500 376500 436500]

rate_of_interest

[nan 4.56 4.25 4.   3.99 4.5   4.125 4.875 3.49 4.375]

Interest_rate_spread

```
[    nan  0.2     0.681   0.3042  0.1523  0.9998  0.2975  0.7395 -0.2776
  0.1871]
```

Upfront_charges

```
[    nan  595.      0.    370.   5120.   5609.88 1150.   2316.5  3953.13
  895.  ]
```

term

[360. 300. 180. 312. 144. 240. 348. 324. 120. 96.]

Neg_ammortization

['not_neg' 'neg_amm' nan]

interest_only

['not_int' 'int_only']

lump_sum_payment

['not_lpsm' 'lpsm']

property_value

```
[ 118000.      nan  508000.  658000.  758000. 1008000.  438000.  308000.
  478000.  688000.]
```

construction_type

```
['sb' 'mh']
```

occupancy_type

```
['pr' 'sr' 'ir']
```

Secured_by

```
['home' 'land']
```

total_units

```
['1U' '2U' '3U' '4U']
```

income

```
[ 1740.  4980.  9480. 11880. 10440. 10080.  5040.  3780.  5580.  6720.]
```

credit_type

```
['EXP' 'EQUI' 'CRIF' 'CIB']
```

Credit_Score

```
[758 552 834 587 602 864 860 863 580 788]
```

co-applicant_credit_type

```
['CIB' 'EXP']
```

age

```
['25-34' '55-64' '35-44' '45-54' '65-74' '>74' '<25' nan]
```

submission_of_application

```
['to_inst' 'not_inst' nan]
```

LTV

```
[98.72881356         nan 80.01968504 69.3768997  91.88654354 70.08928571
 79.10958904 86.52597403 78.76569038 63.44476744]
```

Region

```
['south' 'North' 'central' 'North-East']
```

Security_Type

```
['direct' 'Indriect']
```

Status

```
[1 0]
```

dtir1

```
[45. nan 46. 42. 39. 40. 44. 30. 36. 51.]
```

All columns are well presented in that no numeric column needs to be converted to categorical except for the 'status' column

Drop irrelevant columns (ID and year columns)

```
[ ]: df = df.drop(columns=['ID', 'year'], axis = 1)
```

2.0.2 Rename the columns

```
[ ]: # for all columns in the df, remove any separator or space and replace with "_"
     # All column names should also be in small letters
     def col_rename(df):
         for column in df.columns:
           df.rename(columns={column: column.replace(" ", "_").replace("-", "_").
      ↪lower()}, inplace=True)
         return df

     df = col_rename(df)
```

```
[ ]: df.columns
```

```
[ ]: Index(['loan_limit', 'gender', 'approv_in_adv', 'loan_type', 'loan_purpose',
            'credit_worthiness', 'open_credit', 'business_or_commercial',
            'loan_amount', 'rate_of_interest', 'interest_rate_spread',
            'upfront_charges', 'term', 'neg_ammortization', 'interest_only',
            'lump_sum_payment', 'property_value', 'construction_type',
            'occupancy_type', 'secured_by', 'total_units', 'income', 'credit_type',
            'credit_score', 'co_applicant_credit_type', 'age',
            'submission_of_application', 'ltv', 'region', 'security_type', 'status',
            'dtir1'],
           dtype='object')
```

```
[ ]: df1 = df.copy()
```

Group columns as numeric and categorical

```
[ ]:
```

```
#categorical columns are all columns of object data type, and the status column␣
↪which is numeric
def categorical_numeric_col(df):
    categorical_columns = df.select_dtypes(include=['object']).columns.tolist()

    # Append the 'status' column to categorical columns
    if 'status' in df.columns:
        categorical_columns.append('status')
    # Get all columns in the DataFrame
    all_columns = df.columns
    # Identify numeric columns by excluding categorical columns
    numerical_columns = [col for col in all_columns if col not in␣
↪categorical_columns]

    return categorical_columns, numerical_columns

# Call the function and store results
categorical_columns, numerical_columns = categorical_numeric_col(df)
```

[ ]:
```
# Check the categorical columns
categorical_columns
```

[ ]:
```
['loan_limit',
 'gender',
 'approv_in_adv',
 'loan_type',
 'loan_purpose',
 'credit_worthiness',
 'open_credit',
 'business_or_commercial',
 'neg_ammortization',
 'interest_only',
 'lump_sum_payment',
 'construction_type',
 'occupancy_type',
 'secured_by',
 'total_units',
 'credit_type',
 'co_applicant_credit_type',
 'age',
 'submission_of_application',
 'region',
 'security_type',
 'status']
```

[ ]:
```
# Check the numerical columns
numerical_columns
```

[ ]:
```
['loan_amount',
 'rate_of_interest',
 'interest_rate_spread',
 'upfront_charges',
 'term',
 'property_value',
 'income',
 'credit_score',
 'ltv',
 'dtir1']
```

2.0.3 Handling the missing values

[ ]:
```python
def preprocess_columns(df, numerical_columns, categorical_columns):
    """
    Preprocesses the specified numerical and categorical columns in a DataFrame.

    Parameters:
    - df: The DataFrame to preprocess.
    - numerical_columns: List of numerical columns to impute using KNNImputer.
    - categorical_columns: List of categorical columns to impute using␣
 ↪SimpleImputer.

    Returns:
    - The DataFrame with preprocessed columns.
    """

    # Impute numerical columns using KNNImputer
    if numerical_columns:
        knn_imputer = KNNImputer(n_neighbors=5)
        df[numerical_columns] = knn_imputer.fit_transform(df[numerical_columns])

    # Impute categorical columns using SimpleImputer -- with the mode
    if categorical_columns:
        simple_imputer = SimpleImputer(strategy="most_frequent")
        df[categorical_columns] = simple_imputer.
 ↪fit_transform(df[categorical_columns])

    return df

df = preprocess_columns(df, numerical_columns, categorical_columns)
```

[ ]:
```python
#Check for NAs
df.isna().sum()
```

[ ]:
```
loan_limit                0
gender                    0
```

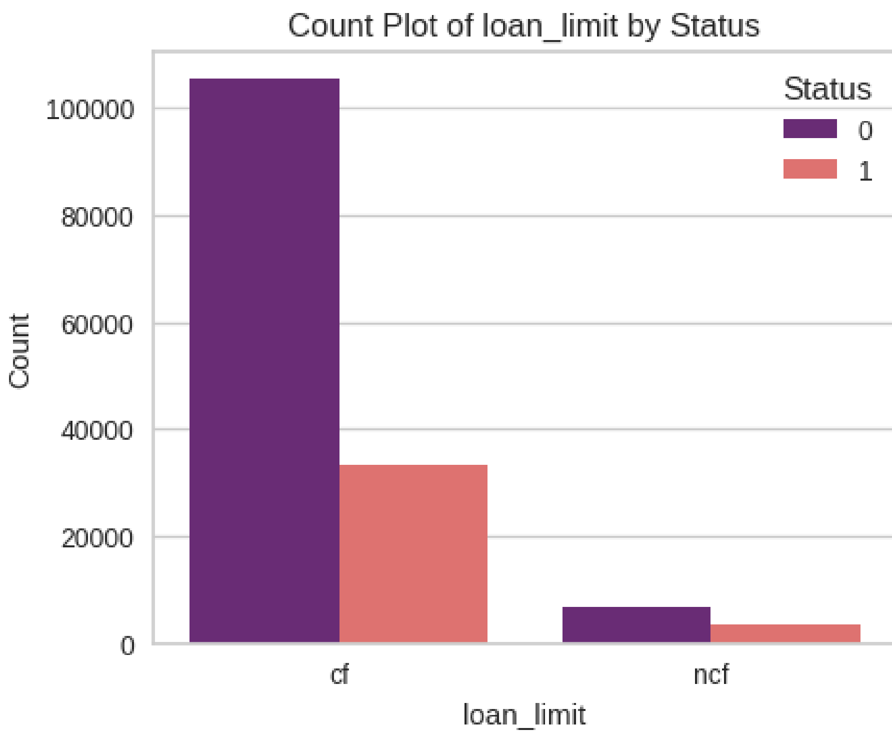
```
approv_in_adv                 0
loan_type                     0
loan_purpose                  0
credit_worthiness             0
open_credit                   0
business_or_commercial        0
loan_amount                   0
rate_of_interest              0
interest_rate_spread          0
upfront_charges               0
term                          0
neg_ammortization             0
interest_only                 0
lump_sum_payment              0
property_value                0
construction_type             0
occupancy_type                0
secured_by                    0
total_units                   0
income                        0
credit_type                   0
credit_score                  0
co_applicant_credit_type      0
age                           0
submission_of_application     0
ltv                           0
region                        0
security_type                 0
status                        0
dtir1                         0
dtype: int64
```

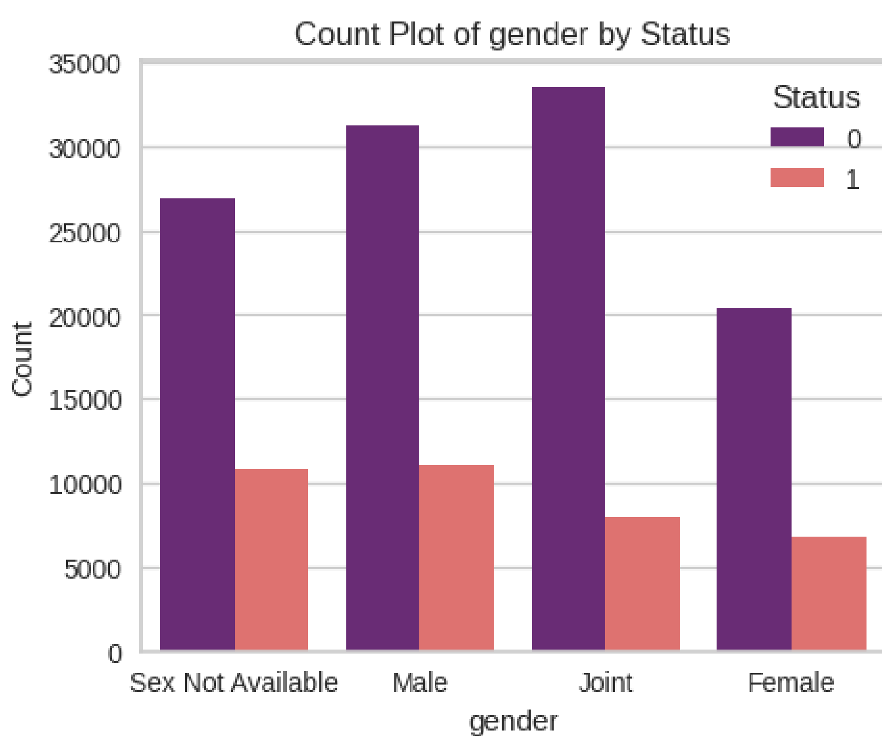
2.1 Step (III): EDA

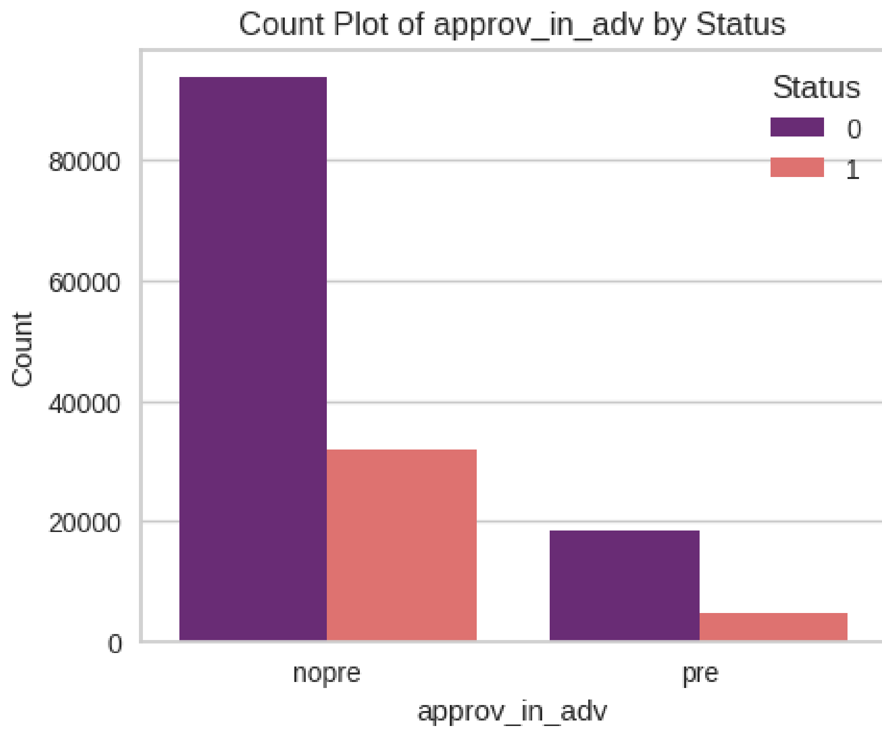
Categorical columns vs Status

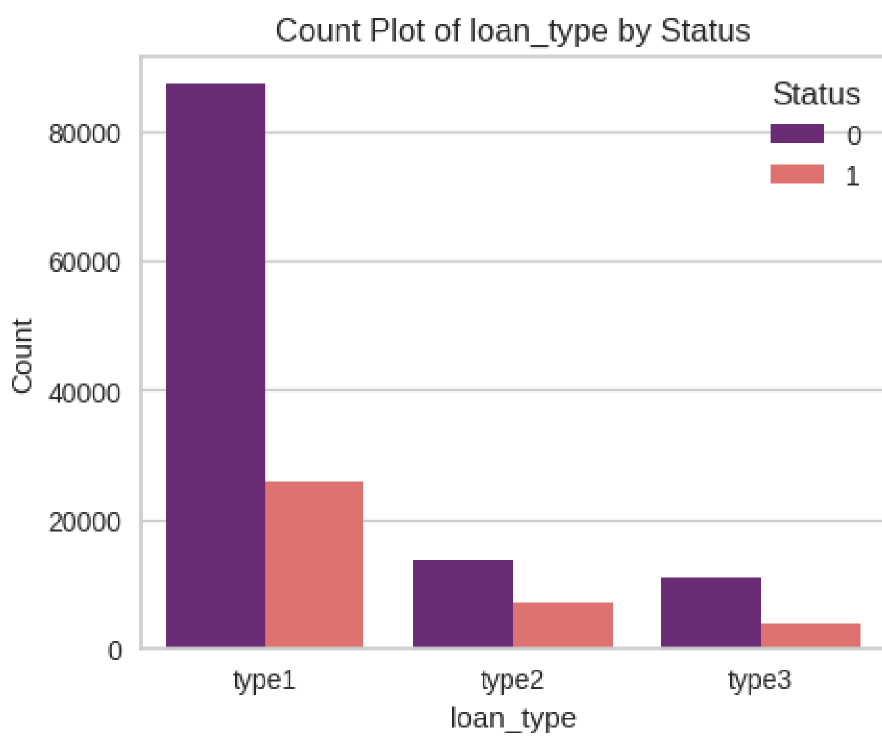
[ ]:
```
#Create count plots
def categorical_vs_status(df, categorical_columns):
    for col in categorical_columns:
        plt.figure(figsize=(5, 4))
        ax = sns.countplot(x=col, hue='status', data=df, palette='magma')
        plt.xlabel(col)
        plt.ylabel('Count')
        plt.title(f'Count Plot of {col} by Status')
        plt.legend(title='Status', loc='upper right')

        plt.show()
```

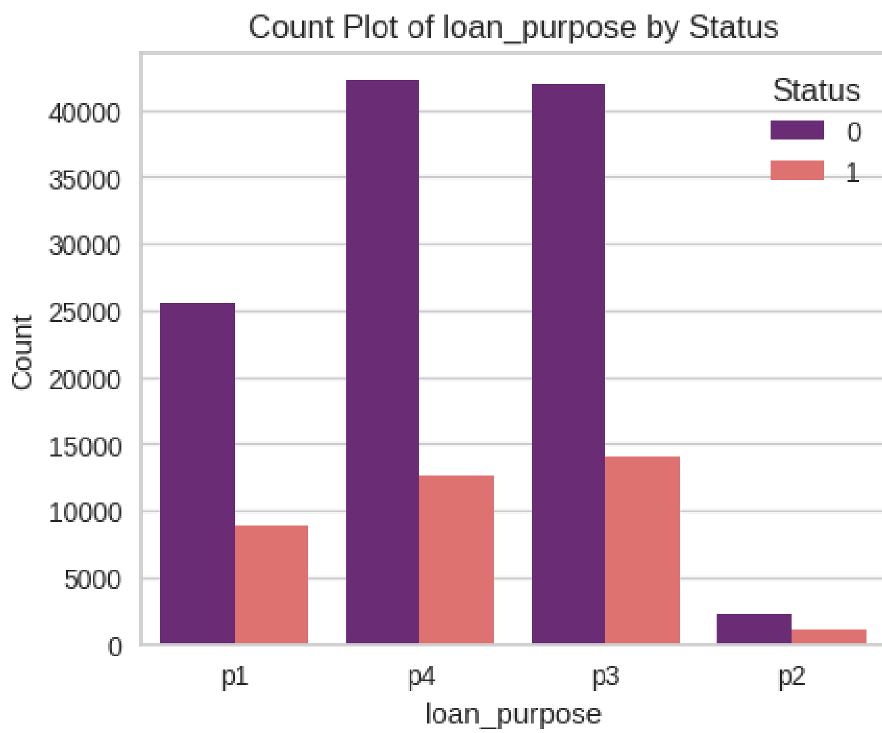
```
categorical_vs_status(df, categorical_columns)
```

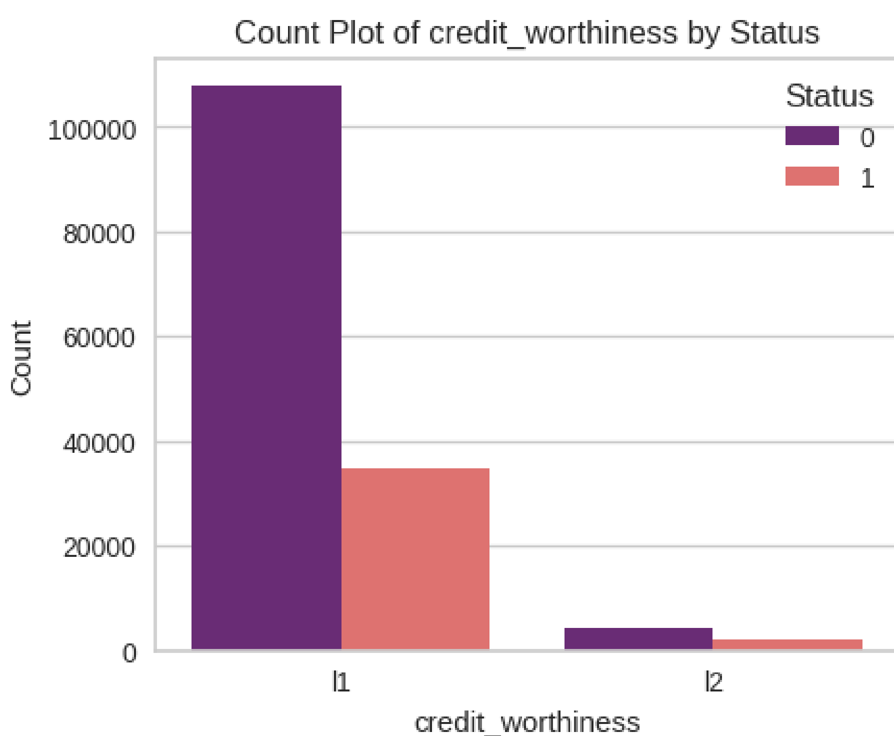


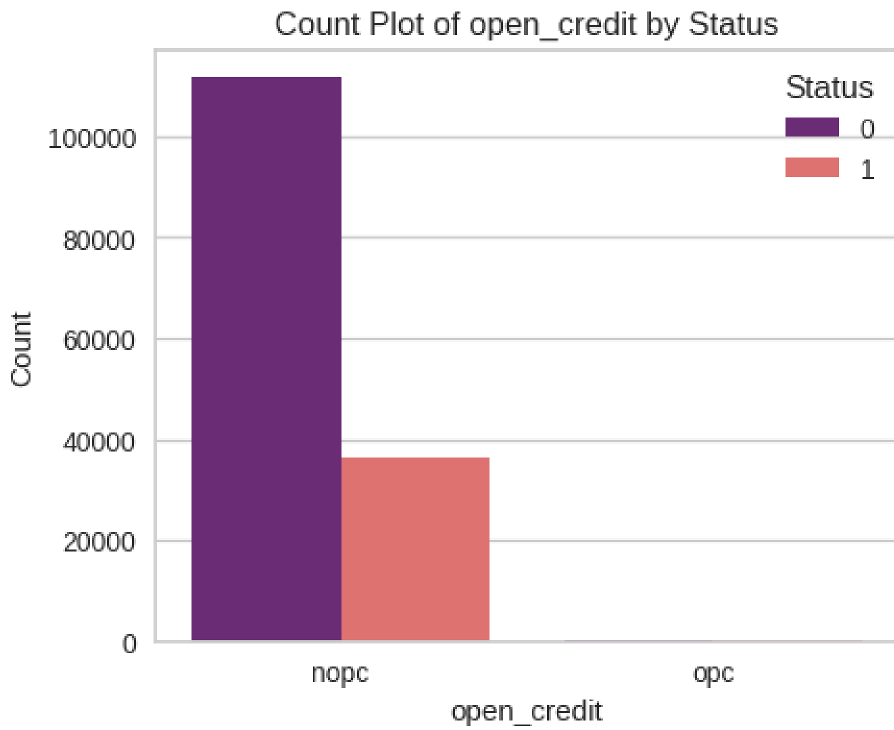


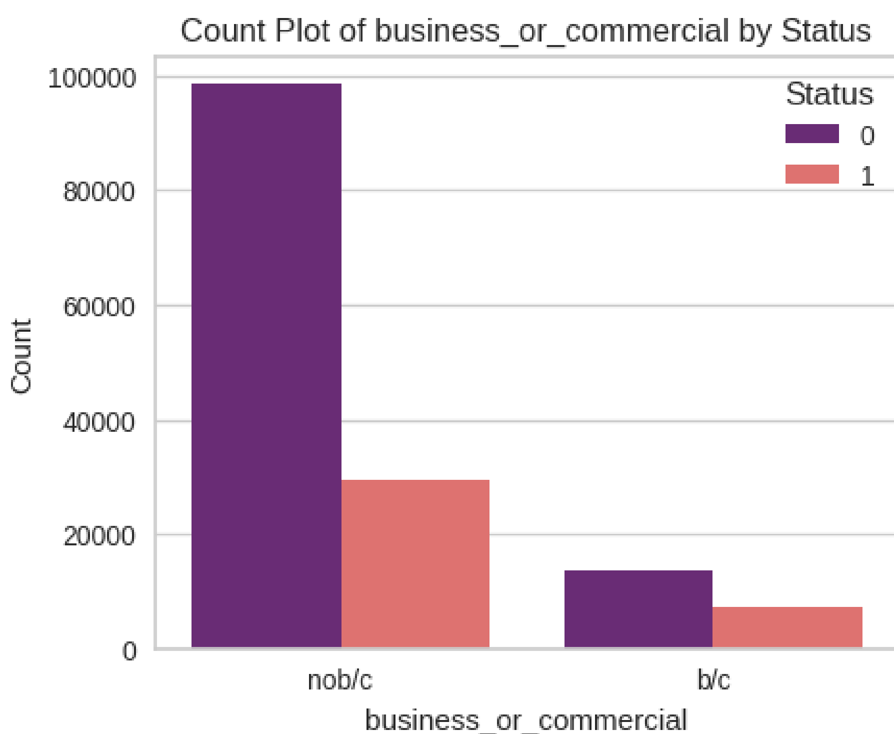

Count Plot of approv_in_adv by Status
Status
0
1
80000
60000
40000
20000
0
Count
nopre
pre
approv_in_adv

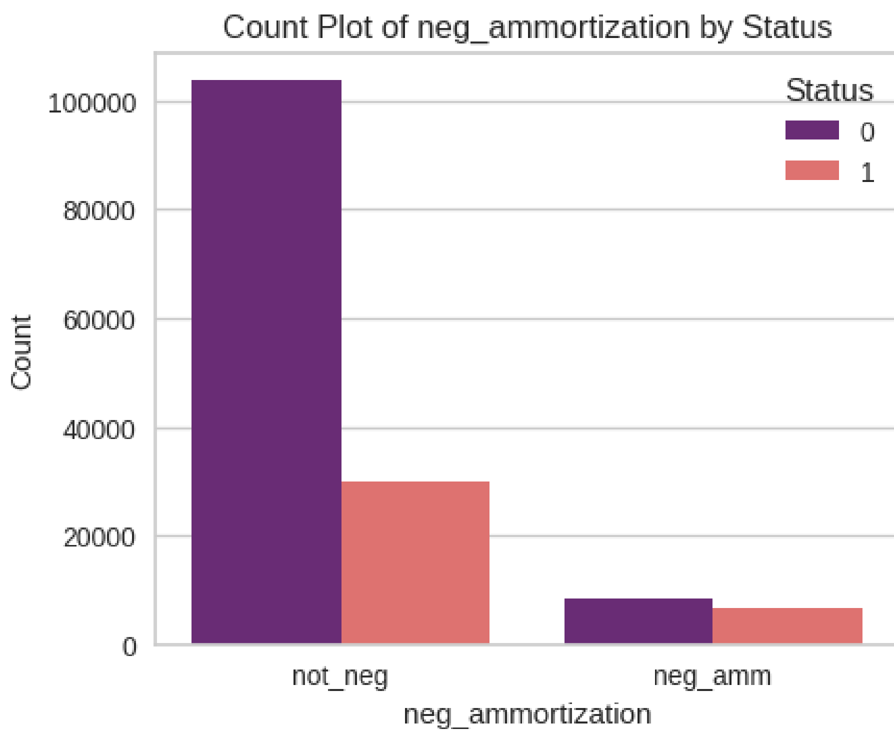

Count Plot of loan_type by Status
Status
0
1
80000
60000
40000
20000
0
Count
type1
type2
type3
loan_type

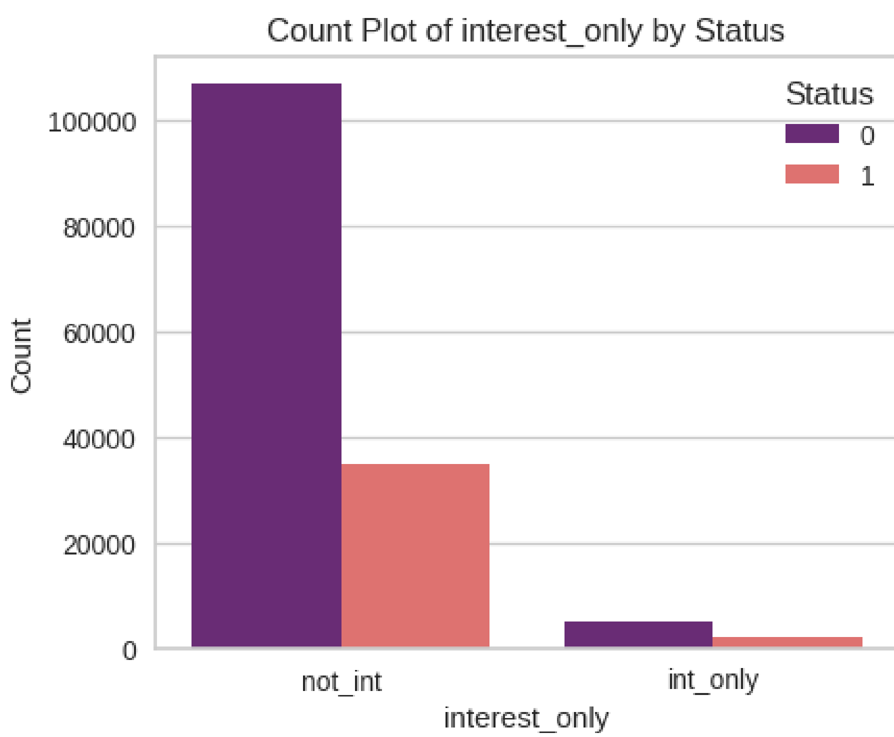

Count Plot of loan_purpose by Status
Status
0
1
40000
35000
30000
25000
20000
15000
10000
5000
0
Count
p1
p4
p3
p2
loan_purpose

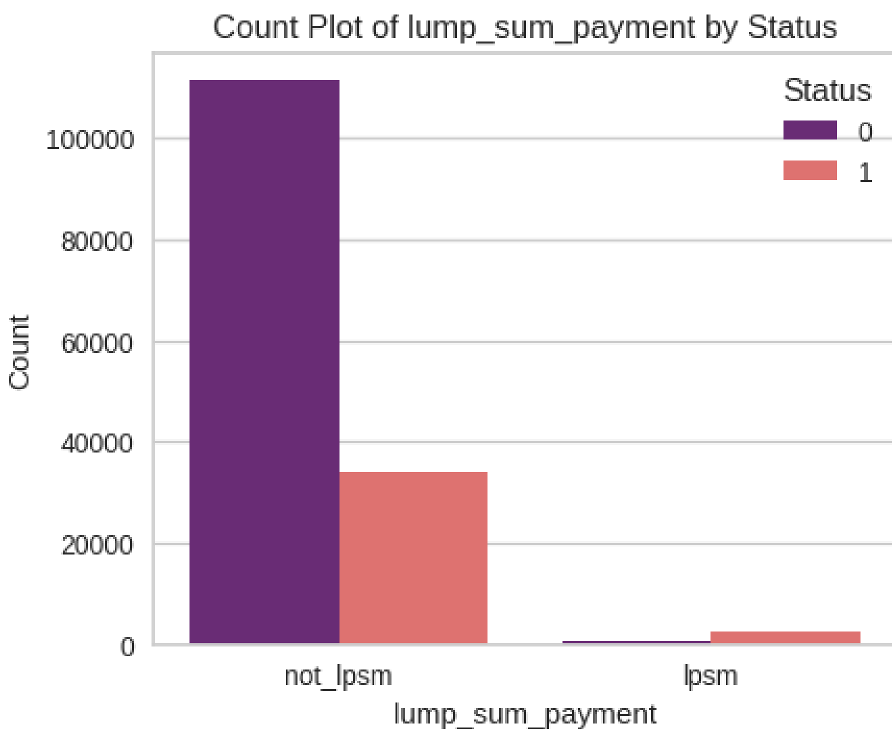

Count Plot of credit_worthiness by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
l1
l2
credit_worthiness

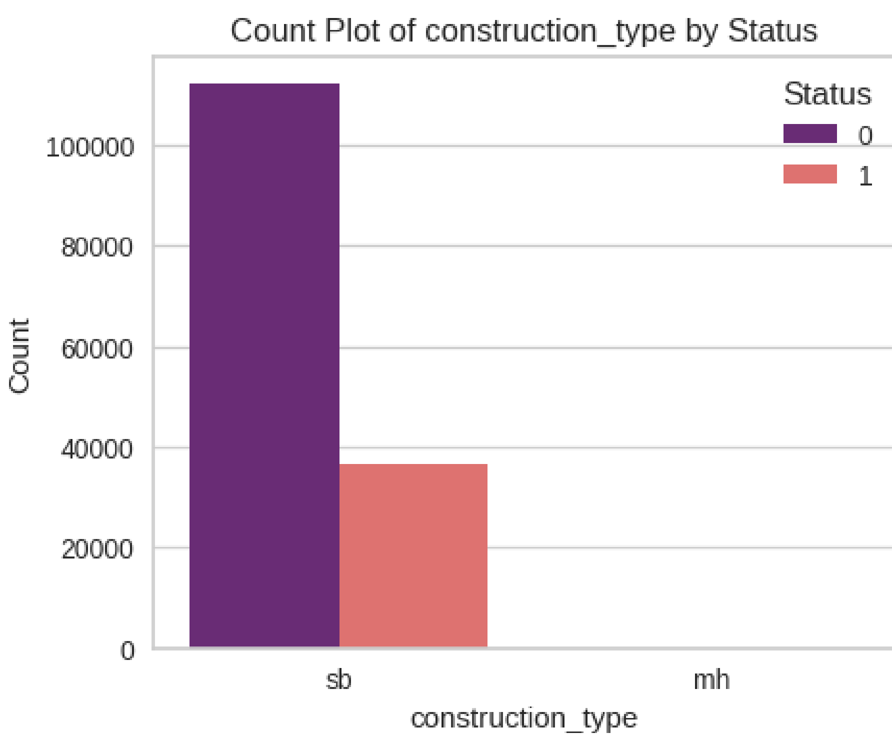

Count Plot of open_credit by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
nopc
opc
open_credit

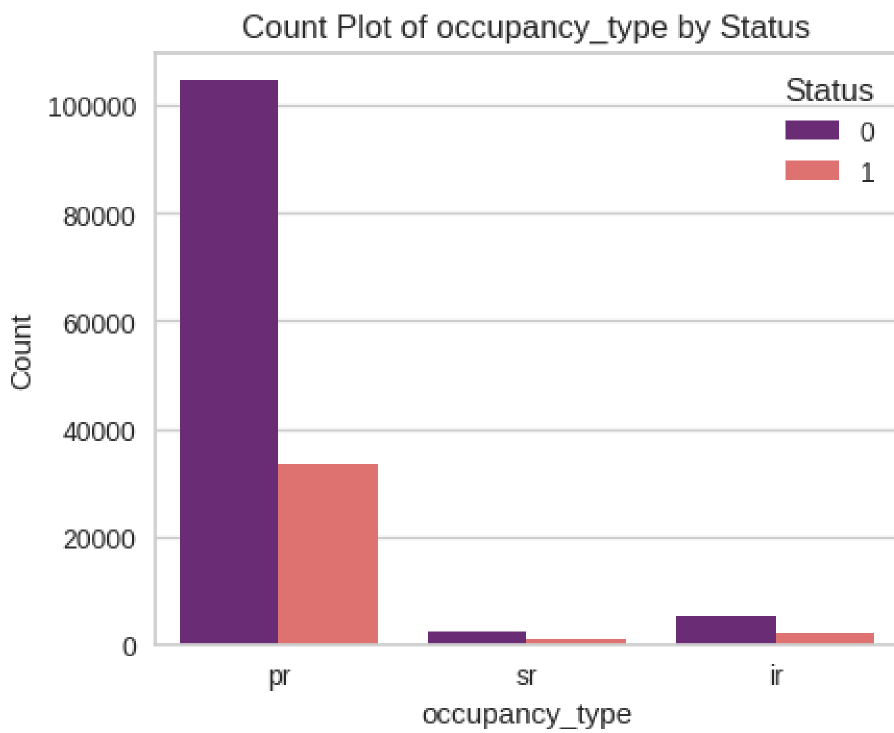

Count Plot of business_or_commercial by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
nob/c
b/c
business_or_commercial

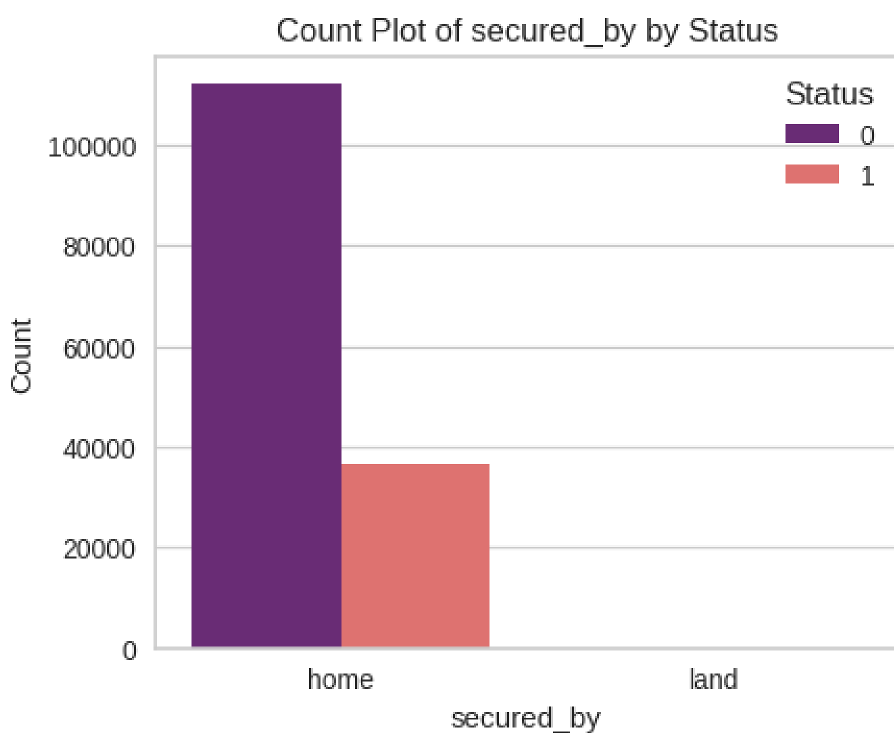

Count Plot of neg_ammortization by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
not_neg
neg_amm
neg_ammortization

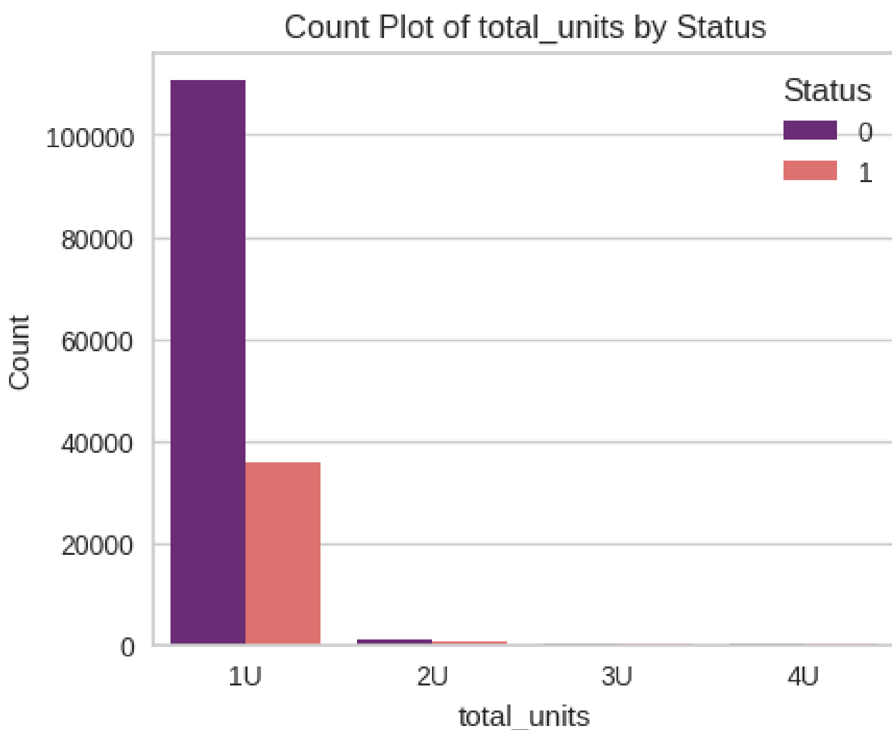

Count Plot of interest_only by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
not_int
int_only
interest_only

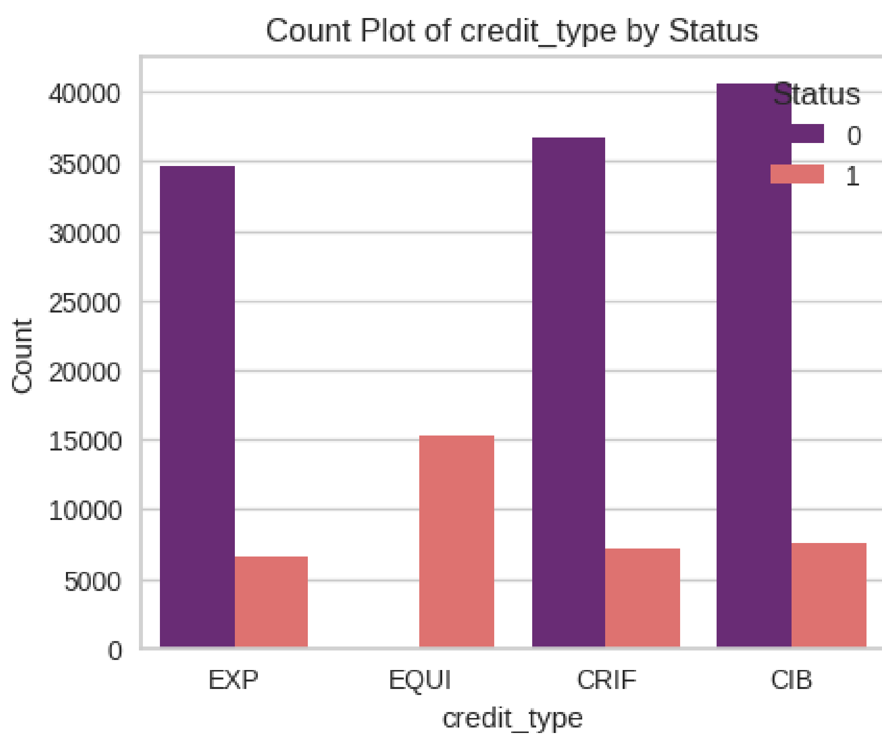

Count Plot of lump_sum_payment by Status
Status
0
1
Count
100000
80000
60000
40000
20000
0
not_lpsm
lpsm
lump_sum_payment

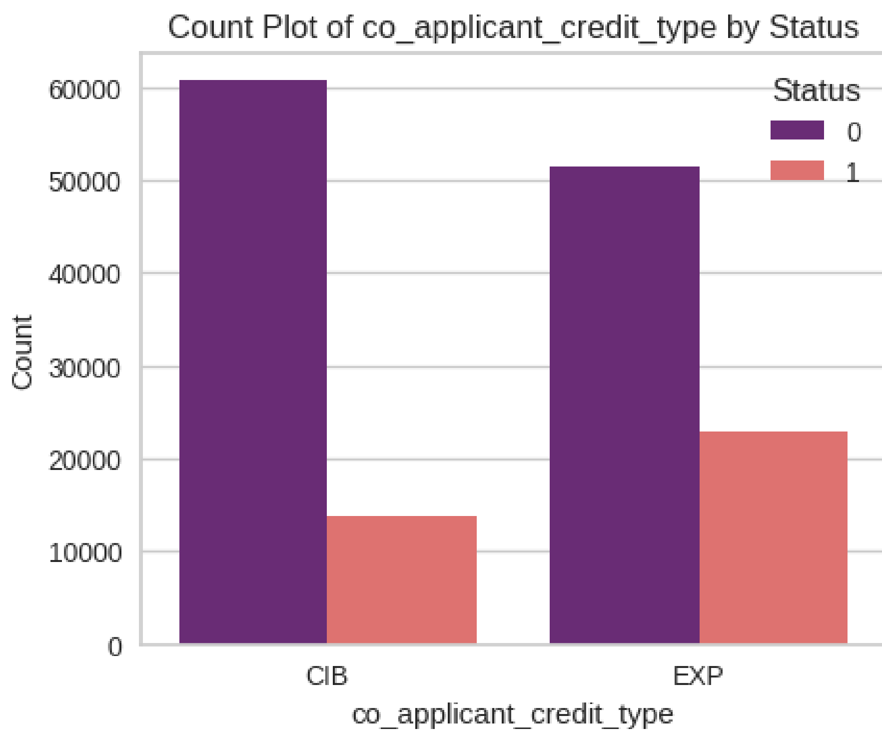

Count Plot of construction_type by Status
Status
0
1
Count
100000
80000
60000
40000
20000
0
sb
mh
construction_type

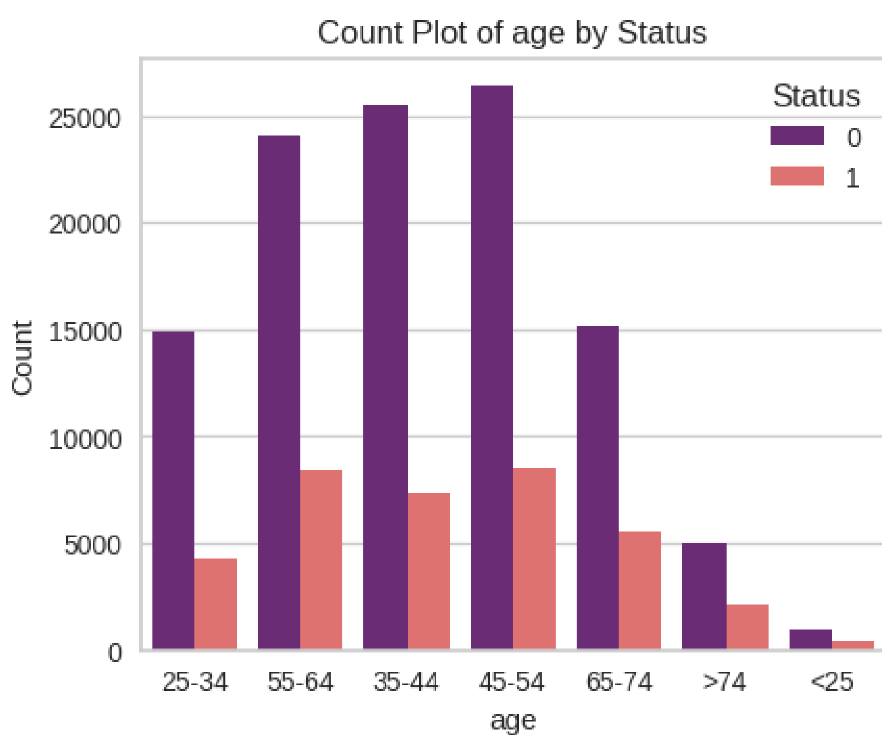

Count Plot of occupancy_type by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
pr
sr
ir
occupancy_type

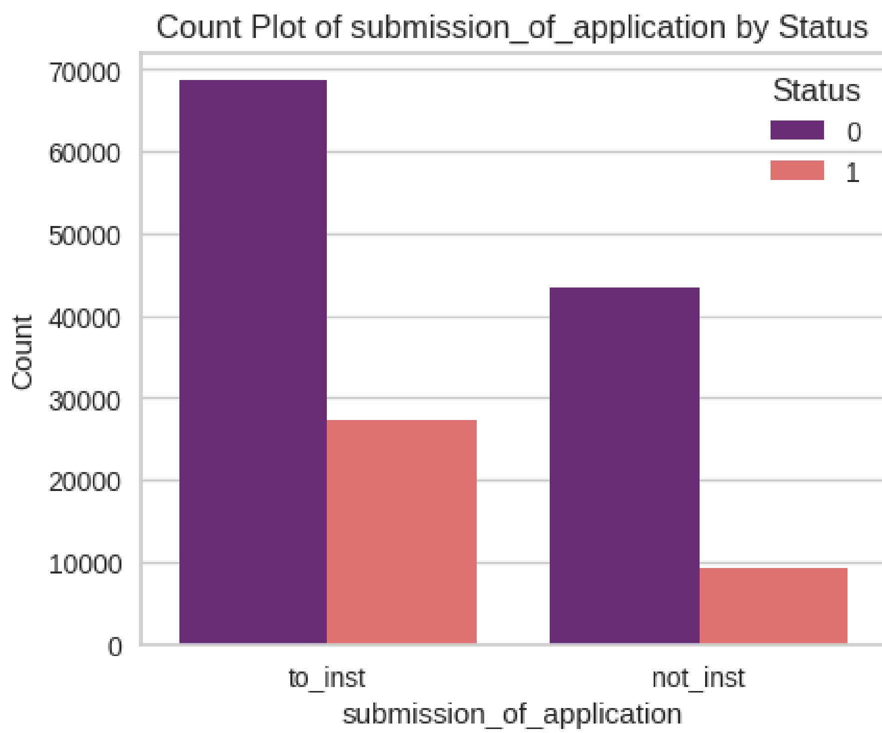

Count Plot of secured_by by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
home
land
secured_by

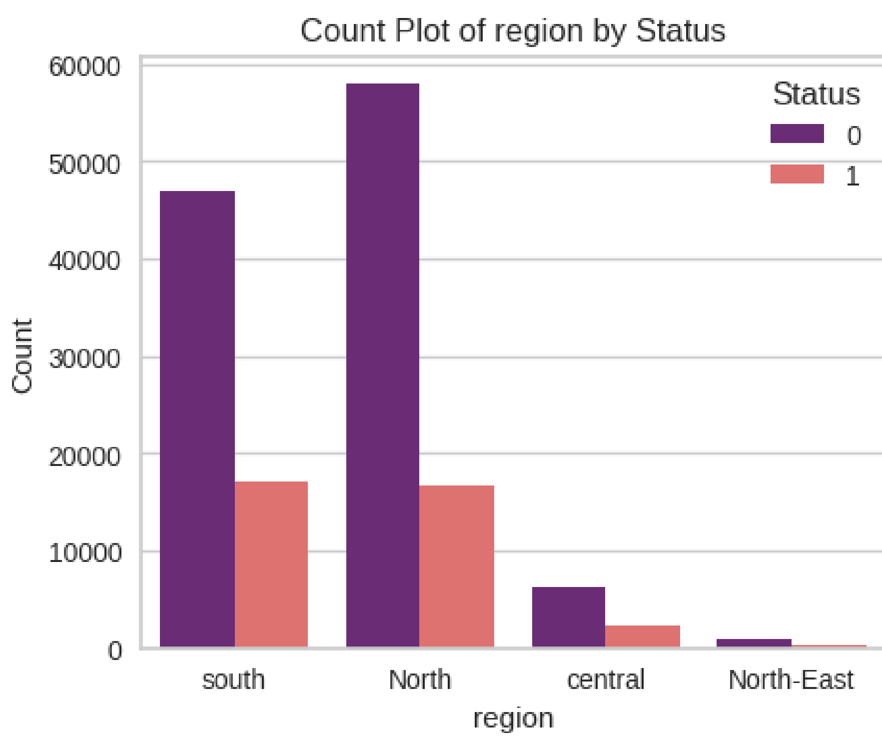

Count Plot of total_units by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
1U
2U
3U
4U
total_units

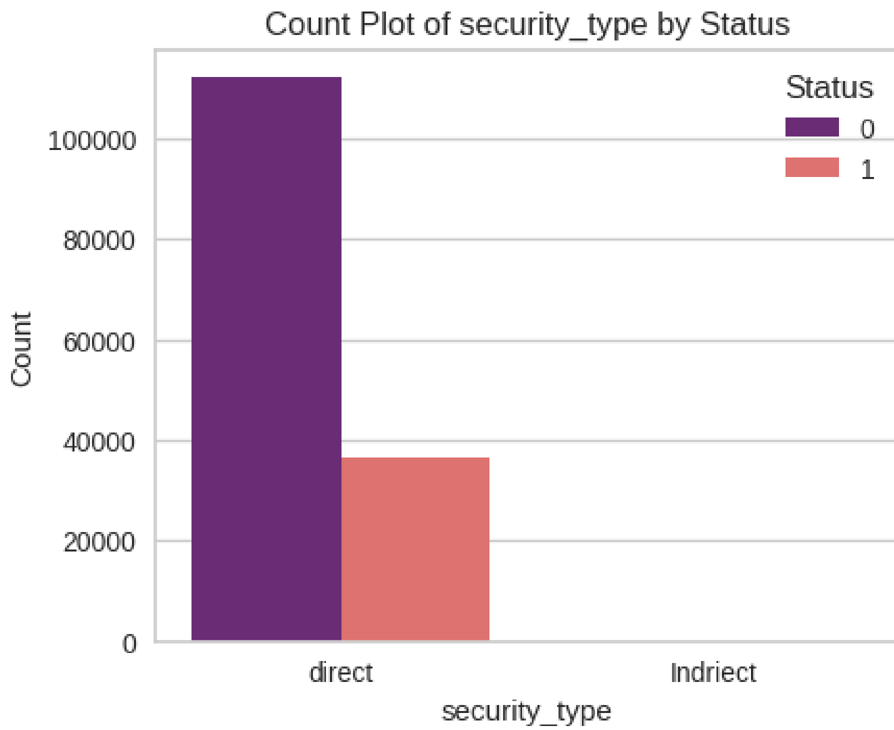

Count Plot of credit_type by Status
Status
0
1
40000
35000
30000
25000
20000
15000
10000
5000
0
Count
EXP
EQUI
CRIF
CIB
credit_type

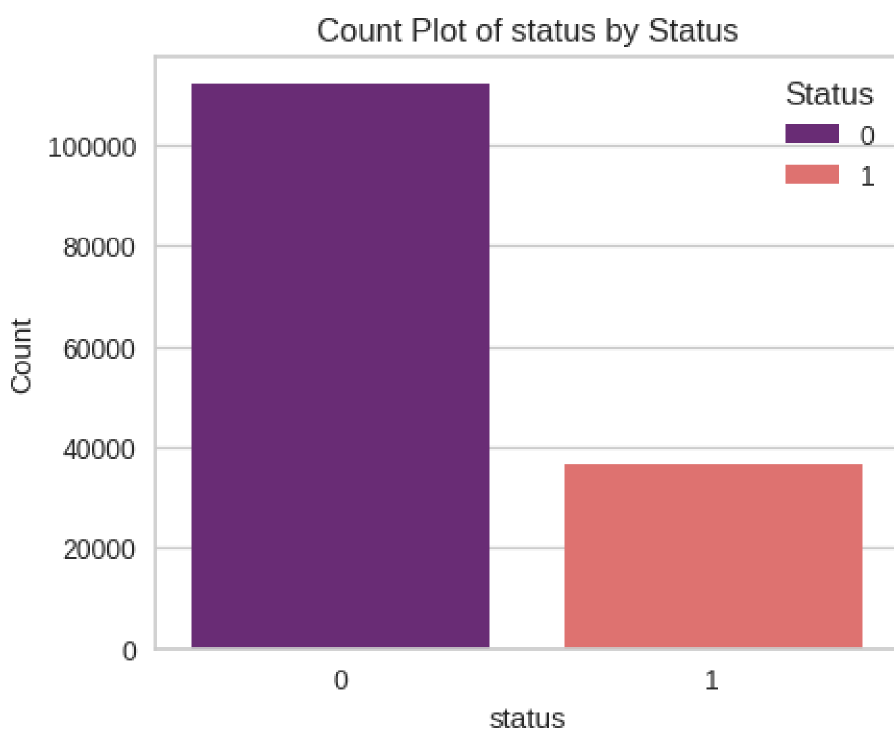

Count Plot of co_applicant_credit_type by Status
Status
0
1
60000
50000
40000
30000
20000
10000
0
Count
CIB
EXP
co_applicant_credit_type


Count Plot of age by Status
Status
0
1
25000
20000
15000
10000
5000
0
Count
25-34
55-64
35-44
45-54
65-74
>74
<25
age


Count Plot of submission_of_application by Status
Count
70000
60000
50000
40000
30000
20000
10000
0
Status
0
1
to_inst
not_inst
submission_of_application


Count Plot of region by Status
Count
60000
50000
40000
30000
20000
10000
0
Status
0
1
south
North
central
North-East
region


Count Plot of security_type by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
direct
Indriect
security_type


Count Plot of status by Status
Status
0
1
100000
80000
60000
40000
20000
0
Count
0
1
status

2.1.1 Comprehensive Report on Loan Default Analysis (Categorical columns)

`Credit Type:`

Analysis of the credit_type column reveals that customers in the 'Equi' category tend to have a higher approval rate, with most of them receiving a status of 1 (approved). In contrast, the 'CIB' category shows the highest level of loan defaults among all credit types, indicating a potential risk group for the lender. Enhanced due diligence or revised lending criteria could be considered for customers in the 'CIB' category to mitigate default risk.

`Loan Limit:`

A significant proportion of loans fall under the 'cf' (confirmed) type of loan limit, demonstrating a preference or need for this loan category. However, the rate of default is lower in 'ncf' (not confirmed) loans compared to 'cf' loans. This suggests that customers who receive 'ncf' loans might be more creditworthy or cautious. The lender might explore increasing the issuance of 'ncf' loans or adjusting terms for 'cf' loans to reduce default rates.

`Gender:`

Analysis by gender reveals that female customers have the lowest default rates, followed by those who do not disclose their gender. Joint borrowers and male customers have the highest loan default rates. This insight indicates a potentially lower risk profile for female borrowers, which could lead to targeted lending campaigns or tailored loan products to attract more female customers, thereby reducing overall default rates.

`Approved in Advance (app_in_adv):`

The data indicates that customers in the 'pre' category (loans approved in advance) have a lower rate of default compared to the 'nopre' category. This suggests that pre-approval processes may effectively filter out higher-risk applicants. Expanding pre-approval procedures could help improve loan performance and reduce default rates.

`Lump Sum Payment:`

Customers opting for the 'lpsm' (lump sum payment) method are almost certain to repay their loans compared to those selecting 'not_lpsm'. This finding suggests that offering or encouraging lump sum payments could significantly reduce default risk. The lender might consider incentivizing lump sum payment options or offering discounts for customers who choose this payment method.

`Age:`

Most customers are concentrated in the 35-44, 45-54, and 55-64 age groups. These groups also have the highest default rates, particularly the 45-54 age group, which leads in loan defaults. Conversely, the '<25' age group has the least number of customers and a notably low default rate. These findings suggest that the lender may benefit from targeting younger demographics who demonstrate lower default risks, while also reassessing the lending terms for middle-aged customers.

`Region:`

The North region has the highest number of customers and also the highest rate of default. Conversely, the Central and North-East regions have the fewest customers and the lowest default rates. The company should consider developing targeted marketing strategies to attract more customers from these low-default regions to improve its risk profile and expand its market share in areas with favourable repayment behaviour.

`Status:`

The data indicates that a large percentage of customers are defaulting on their loans. This high default rate could pose significant financial risk to the lender. Strategic actions, such as refining credit assessment processes, targeting lower-risk demographics, and enhancing risk management protocols, are recommended to mitigate this issue.

2.1.2 Numerical columns

```
[ ]: # Scatter plots to see the distribution of numerical columns and the status as
      ↪the legend
def numeric_vs_status(df):
    y_col = 'loan_amount'
    x_cols = ['rate_of_interest', 'interest_rate_spread', 'upfront_charges',
              'term', 'property_value', 'income', 'credit_score']

    for col in x_cols:
        plt.figure(figsize=(8, 4))
        sns.scatterplot(x=col, y=y_col, data=df, hue='status', palette='magma',
 ↪alpha=0.7)
        # Disable scientific notation
        plt.ticklabel_format(style='plain', axis='both')

        plt.legend(title='Status', loc='upper right')
        plt.xlabel(col)
        plt.ylabel('Loan Amount')
        plt.title(f'Scatter Plot of {col} vs Loan Amount')
        plt.show()

# Apply the function
numeric_vs_status(df)
```

Comprehensive Report on Key Insights from Scatter Plots:

`Rate of Interest:`

When the rate of interest is at 4%, the rate of default is minimal. This suggests that customers are more likely to repay their loans when offered a lower interest rate. Financial institutions may consider maintaining or promoting lower interest rates around this threshold to minimize defaults.

`Interest Rate Spread:`

A spread of 0.5% seems to correlate with a high likelihood of repayment. This implies that keeping the spread narrow — especially around 0.5% — might encourage customers to fulfill their loan obligations. Lower spreads likely make loans more affordable and reduce the burden on borrowers, contributing to improved repayment rates.

`Upfront Charges:`

When upfront charges are around 4,000 (presumably in the relevant currency), customers exhibit a greater likelihood of repaying their loans. It may be beneficial to maintain or standardize these charges around this figure, as it appears to be a manageable upfront cost for many borrowers, enhancing the probability of loan repayment.

`Term Length:`

Loans with terms of 180, 300, and 360 months (15, 25, and 30 years, respectively) show a higher likelihood of repayment. This indicates that customers are more comfortable with these specific loan terms, potentially because they offer flexibility and align with common long-term financial planning. Loan products with these terms could be promoted to improve repayment rates.

`Property Value:`

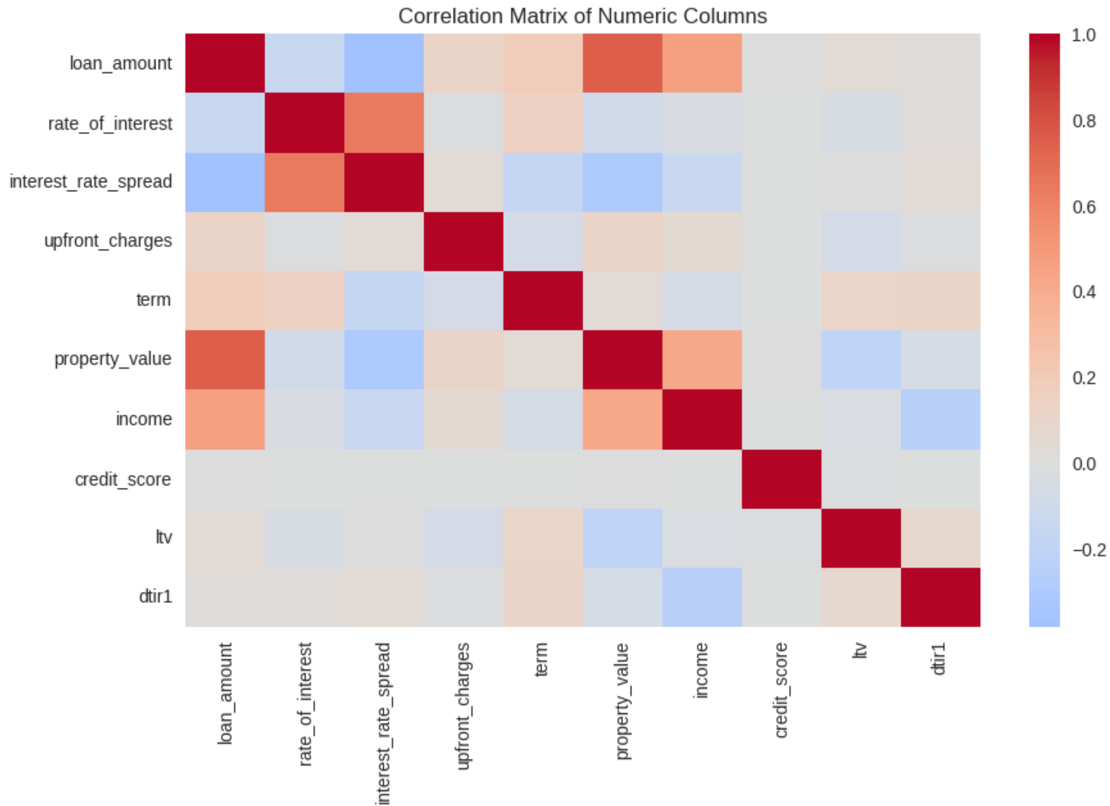
Customers tend to repay loans when the property value is around 500,000 (currency units). In contrast, when property values exceed 2,500,000, and the loan amount is more than 1,500,000, the probability of repayment remains high. This trend suggests that higher-value properties still secure the loans well, perhaps due to their investment stability or the borrowers' stronger financial standing. Marketing strategies targeting high-value properties could thus prove beneficial.

2.1.3 Check for Outliers

[ ]:
```python
def outliers(df):
  for column in df[numerical_columns]:
    plt.figure(figsize=(4,3))
    sns.boxplot(df[column])
    plt.show()

outliers(df)
```

Correlation

[ ]:
```python
# Create correlation heat map
def correlations_df(df):
    numerical_columns = df.select_dtypes(include=['number']).columns
    numeric_df = df[numerical_columns]

    # Plot heatmap for the correlation matrix
    plt.figure(figsize=(10, 6))
    sns.heatmap(numeric_df.corr(), cmap='coolwarm', center=0)
    plt.title('Correlation Matrix of Numeric Columns')
    plt.show()

correlations_df(df)
```



Positive Correlations:

`Loan Amount vs Property Value:`

Higher property values tend to correspond with higher loan amounts, suggesting that more valuable properties are associated with larger loans.

`Loan Amount vs Income:`

Higher income levels are positively correlated with larger loan amounts, indicating that individuals with higher incomes are more likely to qualify for or take out larger loans.

`Interest Rate Spread vs Rate of Interest:`

As the base rate of interest increases, the spread over the base rate also tends to increase, which might indicate a pricing structure based on risk or creditworthiness.

`Income vs Property Value:`

Higher incomes are correlated with higher property values, possibly reflecting that individuals with higher incomes tend to own more valuable properties.

Negative Correlations:

`Interest Rate Spread vs Loan Amount:`

As the loan amount increases, the interest rate spread tends to decrease. This might suggest that larger loans are offered at more competitive rates, possibly due to lower perceived risk or a stronger borrower profile.

`Income vs DTIR1:`

Higher income is associated with a lower debt-to-income ratio, indicating that as income increases, the relative burden of debt payments decreases.

`Status`

The status column, which likely indicates loan approval or default, shows no strong positive or negative correlations with any of the numeric columns. This suggests that loan approval or default may not be directly driven by any single numeric factor, or that the factors influencing status are more complex, possibly involving combinations of several variables or other non-numeric variables.

The lack of strong correlation with status implies that decisions related to loan status (approval or default) involve more intricate criteria, including categorical variables.

[ ]:
```
df.head()
```

Step (IV): Model Building

Encode the categorical variables

[ ]:
```
# drop status from categorical columns
categorical_columns.remove('status')
```

[ ]:
```
categorical_columns
```

[ ]:
```
['loan_limit',
 'gender',
 'approv_in_adv',
 'loan_type',
 'loan_purpose',
 'credit_worthiness',
 'open_credit',
 'business_or_commercial',
 'neg_ammortization',
 'interest_only',
 'lump_sum_payment',
 'construction_type',
 'occupancy_type',
 'secured_by',
 'total_units',
 'credit_type',
 'co_applicant_credit_type',
 'age',
 'submission_of_application',
 'region',
 'security_type']
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['loan_limit', 'gender', 'approv_in_adv', 'loan_type', 'loan_purpose',
       'credit_worthiness', 'open_credit', 'business_or_commercial',
       'loan_amount', 'rate_of_interest', 'interest_rate_spread',
       'upfront_charges', 'term', 'neg_ammortization', 'interest_only',
       'lump_sum_payment', 'property_value', 'construction_type',
       'occupancy_type', 'secured_by', 'total_units', 'income', 'credit_type',
       'credit_score', 'co_applicant_credit_type', 'age',
       'submission_of_application', 'ltv', 'region', 'security_type', 'status',
       'dtir1'],
      dtype='object')
```

[ ]:
```
def encode_categorical(df, categorical_columns):
    """
    One-Hot Encodes the specified categorical columns in a DataFrame.

    Parameters:
    - df: The DataFrame to process.
    - categorical_columns: List of categorical columns to One-Hot Encode.

    Returns:
    - The DataFrame with One-Hot Encoded categorical variables.
    """

    # Apply One-Hot Encoding to categorical columns
    df_encoded = pd.get_dummies(df, columns=categorical_columns,␣
  ↪drop_first=True)

    return df_encoded


df = encode_categorical(df, categorical_columns)
```

Split the data into the dependent(y) and target independent variables (X)

[ ]:
```
# Separate input features (X) and target variable (y)
X = df.drop(columns='status')
y = df['status']

# Split the 2 into train and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
  ↪random_state=42)

print(f"The x shape is:, {X.shape}")
print(f"\n The y shape is:, {y.shape}")
print(f"\n The x_train shape is:, {X_train.shape}")
print(f"\n The x_test shape is:, {X_test.shape}")
```

```
print(f"\n The y_train shape is:, {y_train.shape}")
print(f"\n The y_test shape is:, {y_test.shape}")
```

```
The x shape is:, (148670, 48)

 The y shape is:, (148670,)

 The x_train shape is:, (118936, 48)

 The x_test shape is:, (29734, 48)

 The y_train shape is:, (118936,)

 The y_test shape is:, (29734,)
```

[ ]:
```
y_train = y_train.astype(int)
y_test = y_test.astype(int)
```

[ ]:
```
X_train.head(3)
```

[ ]:
```
        loan_amount  rate_of_interest  interest_rate_spread  upfront_charges  \
141245      76500.0             3.500                0.0551         2625.000
3507       556500.0             4.000                0.1255         5801.776
53688      126500.0             3.625                1.4909         3157.580

         term  property_value  income  credit_score        ltv  dtir1  …  \
141245  360.0        108000.0  2460.0         605.0  70.833333   12.0  …
3507    360.0        928000.0  7200.0         729.0  59.967672   43.0  …
53688   180.0        148000.0  2100.0         609.0  85.472973   42.0  …

        age_45-54  age_55-64  age_65-74  age_<25  age_>74  \
141245      False      False      False    False     True
3507         True      False      False    False    False
53688       False      False       True    False    False

        submission_of_application_to_inst  region_North-East  region_central  \
141245                               True              False           False
3507                                False              False           False
53688                                True              False           False

        region_south  security_type_direct
141245          True                  True
3507            True                  True
53688          False                  True

[3 rows x 48 columns]
```

[ ]:
```
y_train.head
```

[ ]:
```
<bound method NDFrame.head of 141245    0
3507      0
53688     0
46491     1
54671     0
         ..
119879    1
103694    0
131932    0
146867    0
121958    1
Name: status, Length: 118936, dtype: object>
```

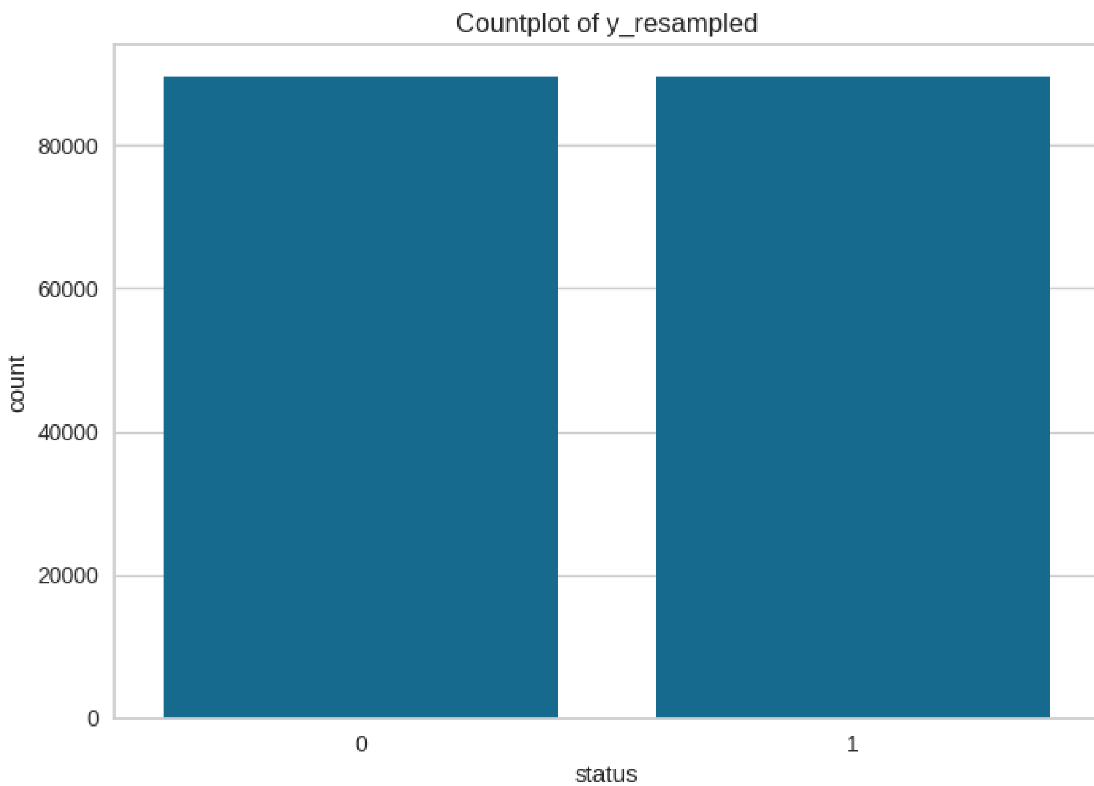
2.1.4 Resampling

Since there is an imbalance in the target variable i.e there is an imbalance in the status values (0 and 1), I use Synthetic Minority Over-sampling Technique (SMOTE) to generates synthetic samples based on the feature space similarities between existing minority instances.

[ ]:
```python
def apply_smote(X_train, y_train):
      """
      Applies SMOTE to balance the dataset based on the target column.

      Parameters:
      df (pd.DataFrame): The input DataFrame containing features and the target␣
 ↪column.
      target_column (str): The name of the target column to balance.

      Returns:
      X_train_resampled (pd.DataFrame): The resampled input features.
      y_train_resampled (pd.Series): The resampled target variable.
      """


      # Apply SMOTE to balance the dataset
      smote = SMOTE(random_state=42)
      X_train_resampled, y_train_resampled = smote.fit_resample(X_train,␣
 ↪y_train)

      # Convert resampled arrays back to DataFrame and Series
      X_train_resampled = pd.DataFrame(X_train_resampled, columns=X_train.
 ↪columns)
      y_train_resampled = pd.Series(y_train_resampled, name='status')

      return X_train_resampled, y_train_resampled
```

```
X_train_resampled, y_train_resampled = apply_smote(X_train, y_train)
```

[ ]:
```
# Create a countplot to show the y_sampled values
sns.countplot(x=y_train_resampled)
plt.title('Countplot of y_resampled')
plt.show()
```



[ ]:
```
# Rename the sampled data back to X and y train
X_train = X_train_resampled
y_train = y_train_resampled
```

[ ]:
```
print(f"The x shape is:, {X.shape}")
print(f"\n The y shape is:, {y.shape}")
print(f"\n The x_train shape is:, {X_train.shape}")
print(f"\n The x_test shape is:, {X_test.shape}")
print(f"\n The y_train shape is:, {y_train.shape}")
print(f"\n The y_test shape is:, {y_test.shape}")
```

```
The x shape is:, (148670, 48)

 The y shape is:, (148670,)
```

```
The x_train shape is:, (179074, 48)

The x_test shape is:, (29734, 48)

The y_train shape is:, (179074,)

The y_test shape is:, (29734,)
```

using pycaret to choose the best performing models

[ ]:
```
setup(data=pd.concat([X_train, y_train], axis=1), target='status')
```

```
<pandas.io.formats.style.Styler at 0x7d6254555030>
```

[ ]:
```
<pycaret.classification.oop.ClassificationExperiment at 0x7d6254f75000>
```

[ ]:
```
compare_models()
```

```
<IPython.core.display.HTML object>

<pandas.io.formats.style.Styler at 0x7d62552df310>

Processing:   0%|          | 0/65 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[ ]:
```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
               random_state=5893, reg_alpha=0.0, reg_lambda=0.0, subsample=1.0,
               subsample_for_bin=200000, subsample_freq=0)
```

2.1.5 Function to make predictions based on 3 models, (Decision trees, Random Forest, and XGBoost)

[ ]:
```
#Function to make the predictions and output a classification report
def apply_classifiers(X_train, X_test, y_train, y_test):
      # Decision Tree Classifier
      dt = DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,␣
 ↪criterion='gini',
                        max_depth=None, max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_samples_leaf=1,
                        min_samples_split=2, min_weight_fraction_leaf=0.0,
                        monotonic_cst=None, random_state=7215, splitter='best')

      #Random forest
      rf = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,␣
 ↪class_weight=None,
                                   criterion='gini', max_depth=None,␣
 ↪max_features='log2',
```

```
                                  min_impurity_decrease=0.0, min_samples_leaf=1,
                                  min_samples_split=2, n_estimators=100,
                                  random_state=7215, verbose=0)
    # XGBoost Classifier
    xgb_model = xgb.XGBClassifier(random_state=42)
    #Ada boost classifier
    ada_model = AdaBoostClassifier(random_state = 42)
    #LGBM Classifier
    lgbm_model = LGBMClassifier(boosting_type='gbdt', class_weight=None,␣
↪colsample_bytree=1.0,
                    importance_type='split', learning_rate=0.1, max_depth=-1,
                    min_child_samples=20, min_child_weight=0.001,␣
↪min_split_gain=0.0,
                    n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
                    random_state=5893, reg_alpha=0.0, reg_lambda=0.0,␣
↪subsample=1.0,
                    subsample_for_bin=200000, subsample_freq=0)



    # Map model names to model objects
    models = {'Decision Tree': dt, 'Random Forest': rf, 'XGBoost': xgb_model,␣
↪'AdaBoost': ada_model, 'Light Gradient Boosting Machine': lgbm_model}

    trained_models = {}
    # Ensure all column names are strings and remove problematic characters
    X_train.columns = X_train.columns.astype(str).str.replace('[', '').str.
↪replace(']', '').str.replace('<', '')
    X_test.columns = X_test.columns.astype(str).str.replace('[', '').str.
↪replace(']', '').str.replace('<', '')



    for model_name, model in models.items():
        # Train the model
        model.fit(X_train, y_train)
        trained_models[model_name] = model
        # Make predictions
        y_pred = model.predict(X_test)

        # Print classification report
        print(f"\n Classification Report for {model_name}: \n")
        print(classification_report(y_test, y_pred))
```

```
    return trained_models

trained_models = apply_classifiers(X_train, X_test, y_train, y_test)
trained_models
```

```
Classification Report for Decision Tree:

              precision    recall  f1-score   support

           0       0.97      0.96      0.97     22494
           1       0.89      0.91      0.90      7240

    accuracy                           0.95     29734
   macro avg       0.93      0.94      0.93     29734
weighted avg       0.95      0.95      0.95     29734


Classification Report for Random Forest:

              precision    recall  f1-score   support

           0       0.94      0.98      0.96     22494
           1       0.93      0.80      0.86      7240

    accuracy                           0.94     29734
   macro avg       0.94      0.89      0.91     29734
weighted avg       0.94      0.94      0.94     29734


Classification Report for XGBoost:

              precision    recall  f1-score   support

           0       0.99      1.00      0.99     22494
           1       0.99      0.96      0.98      7240

    accuracy                           0.99     29734
   macro avg       0.99      0.98      0.98     29734
weighted avg       0.99      0.99      0.99     29734


Classification Report for AdaBoost:

              precision    recall  f1-score   support

           0       0.92      0.93      0.93     22494
           1       0.77      0.76      0.76      7240
```

```
    accuracy                           0.89     29734
   macro avg       0.85      0.84      0.84     29734
weighted avg       0.89      0.89      0.89     29734

[LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
[LightGBM] [Info] Number of positive: 89537, number of negative: 89537
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.070903 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 2615
[LightGBM] [Info] Number of data points in the train set: 179074, number of used
features: 48
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000

 Classification Report for Light Gradient Boosting Machine:

              precision    recall  f1-score   support

           0       0.99      1.00      0.99     22494
           1       0.99      0.96      0.98      7240

    accuracy                           0.99     29734
   macro avg       0.99      0.98      0.98     29734
weighted avg       0.99      0.99      0.99     29734
```

[ ]:

```
{'Decision Tree': DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
criterion='gini',
                        max_depth=None, max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_samples_leaf=1,
                        min_samples_split=2, min_weight_fraction_leaf=0.0,
                        monotonic_cst=None, random_state=7215, splitter='best'),
 'Random Forest': RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
class_weight=None,
                        criterion='gini', max_depth=None, max_features='log2',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_samples_leaf=1,
                        min_samples_split=2, min_weight_fraction_leaf=0.0,
                        monotonic_cst=None, n_estimators=100, n_jobs=None,
                        oob_score=False, random_state=7215, verbose=0,
                        warm_start=False),
 'XGBoost': XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
```

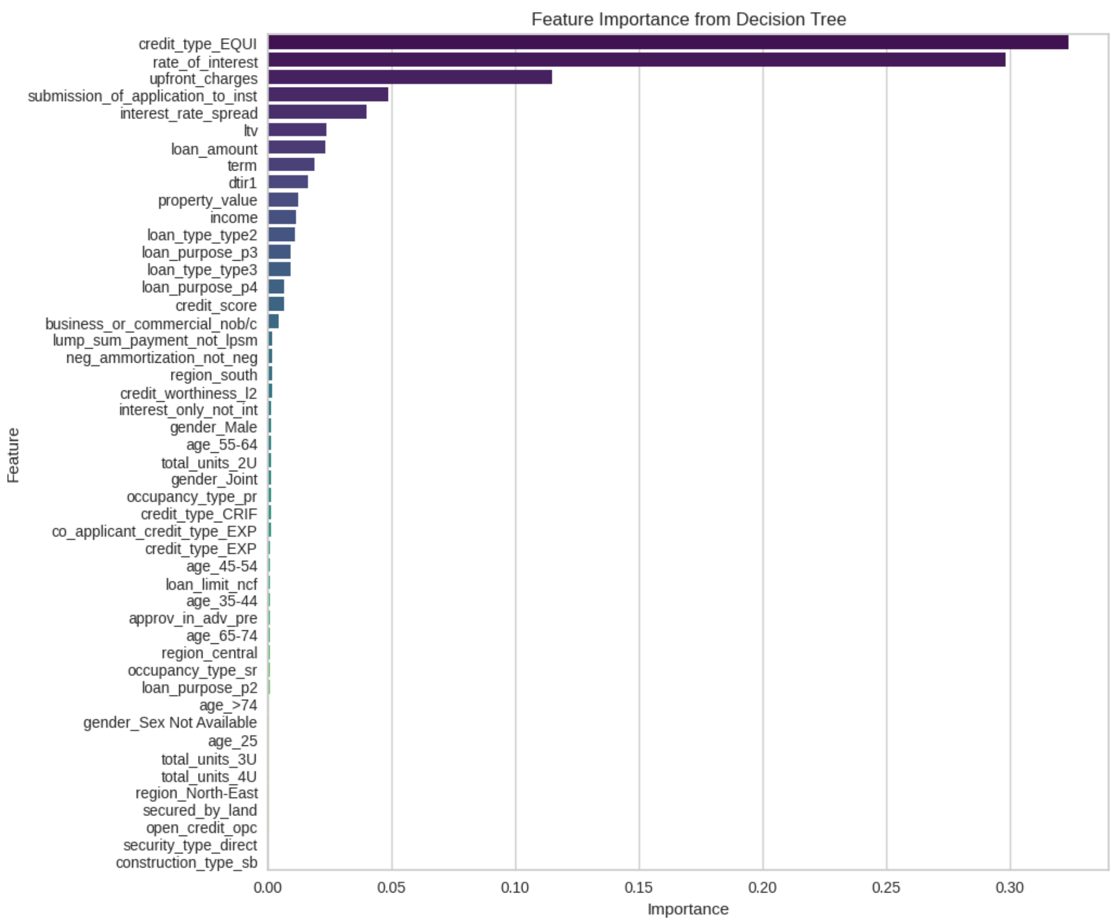
```
                gamma=None, grow_policy=None, importance_type=None,
                interaction_constraints=None, learning_rate=None, max_bin=None,
                max_cat_threshold=None, max_cat_to_onehot=None,
                max_delta_step=None, max_depth=None, max_leaves=None,
                min_child_weight=None, missing=nan, monotone_constraints=None,
                multi_strategy=None, n_estimators=None, n_jobs=None,
                num_parallel_tree=None, objective='binary:logistic', …),
 'AdaBoost': AdaBoostClassifier(algorithm='SAMME.R', estimator=None,
learning_rate=1.0,
                    n_estimators=50, random_state=42),
 'Light Gradient Boosting Machine': LGBMClassifier(boosting_type='gbdt',
class_weight=None, colsample_bytree=1.0,
                importance_type='split', learning_rate=0.1, max_depth=-1,
                min_child_samples=20, min_child_weight=0.001,
min_split_gain=0.0,
                n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
                random_state=5893, reg_alpha=0.0, reg_lambda=0.0, subsample=1.0,
                subsample_for_bin=200000, subsample_freq=0)}
```

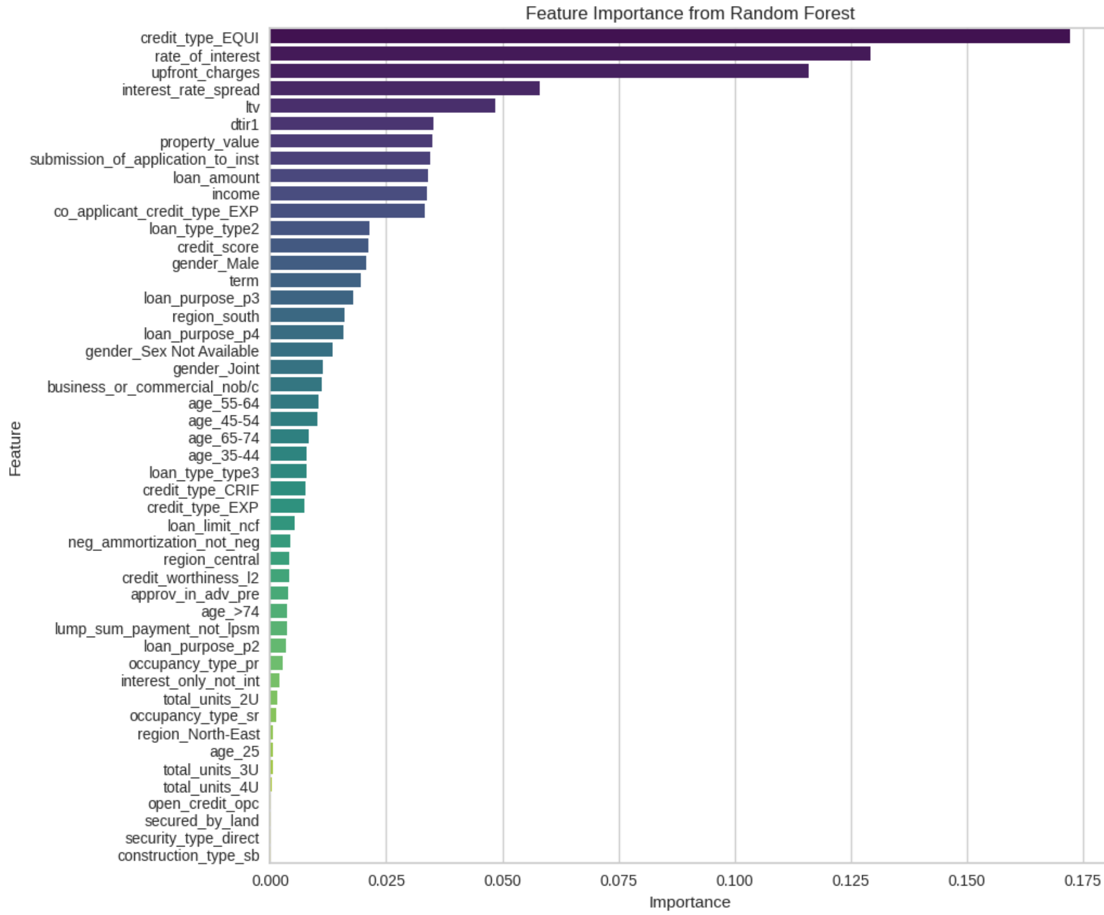
All five models—Decision Tree, Random Forest, XGBoost, AdaBoost, and LightGBM—show exceptional performance with perfect classification metrics. The models show varying performance, with XGBoost and LightGBM achieving the highest accuracy (99%), precision, recall, and f1-scores across both classes. Decision Tree performs well with 95% accuracy, while AdaBoost lags behind at 89%. XGBoost or LightGBM is the best due to their superior overall metrics, especially for class 1.

Get feature importance

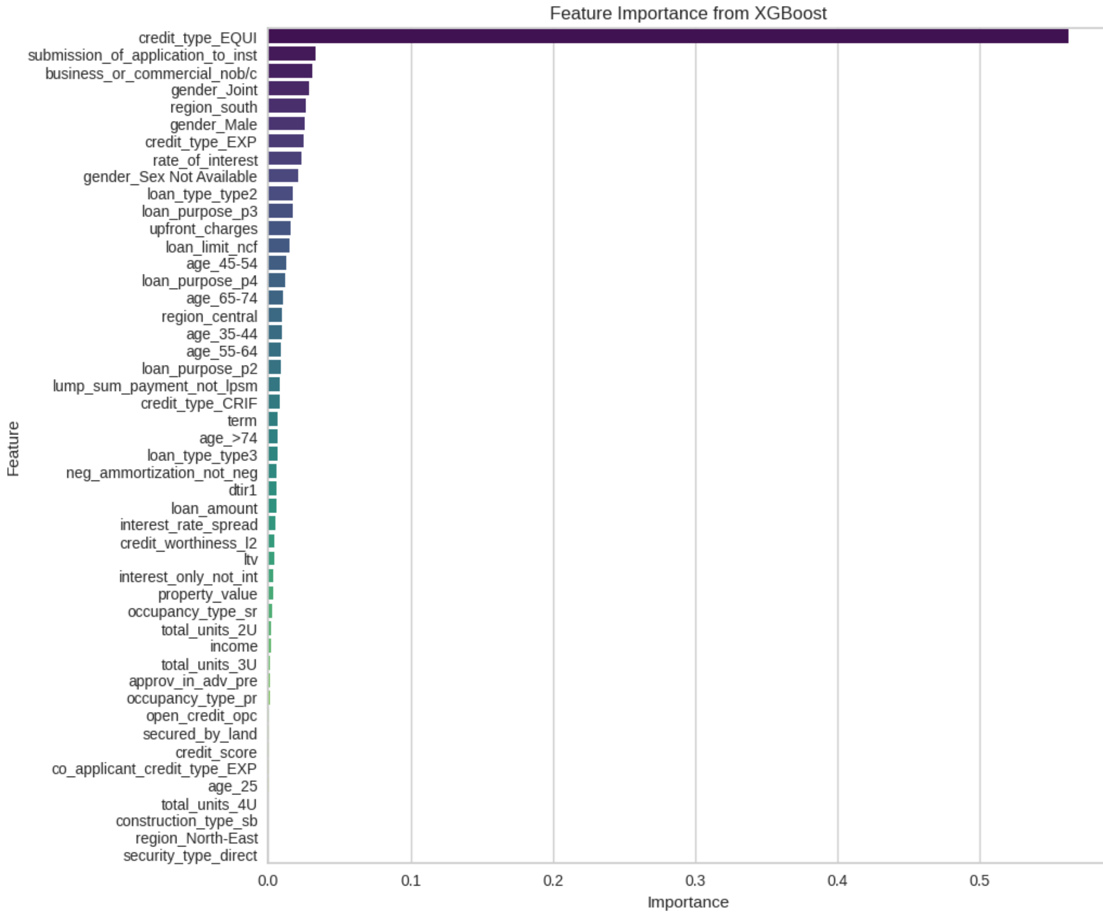
[ ]:
```python
# Feature importance function

def plot_feature_importances(models, X):
    """
    Plots the feature importance of multiple trained models.

    Parameters:
    models (dict): Dictionary of model names and trained models.
    X (pd.DataFrame): The input features used to train the models.
    """
    for model_name, model in models.items():
        # Check if the model has the feature_importances_ attribute
        if hasattr(model, 'feature_importances_'):
            importances = model.feature_importances_
            features = X.columns

            # Create a DataFrame for plotting
            feature_importance_df = pd.DataFrame({
                'Feature': features,
                'Importance': importances
```

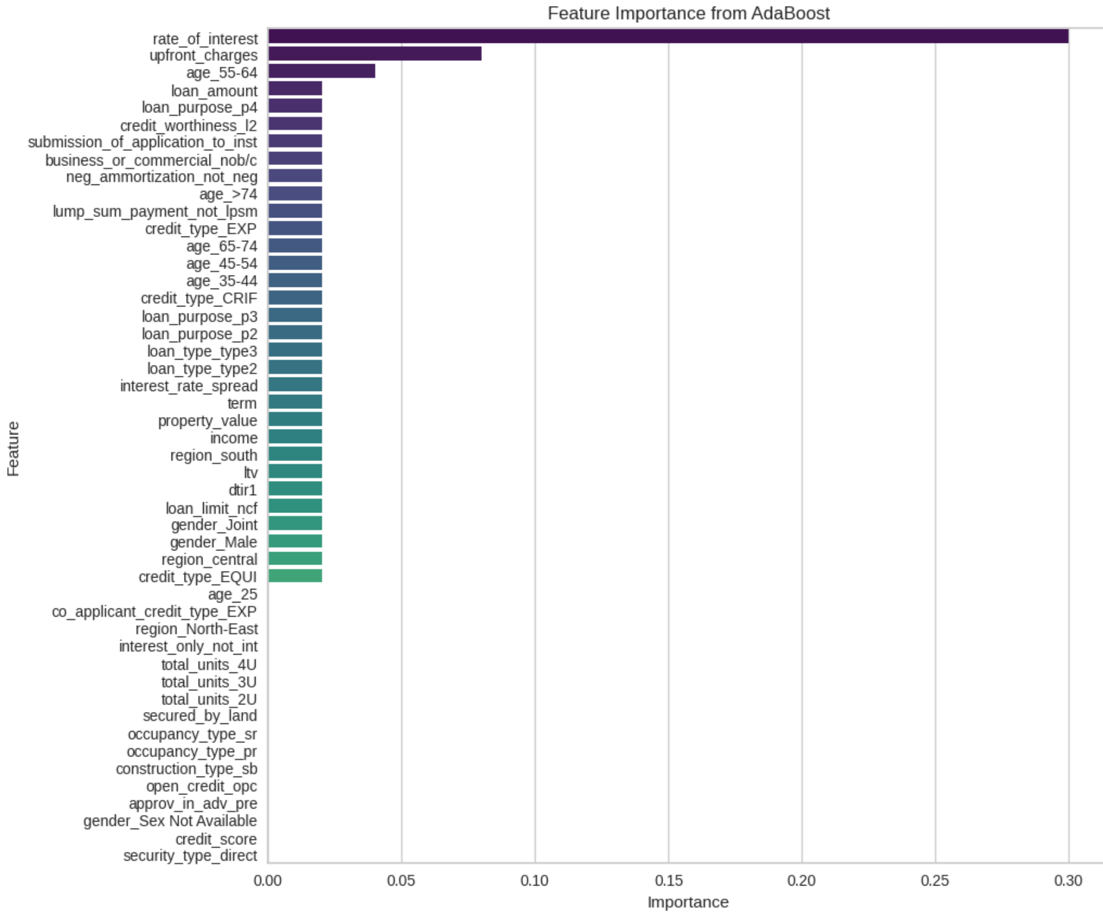
```
            }).sort_values(by='Importance', ascending=False)

            # Plot
            plt.figure(figsize=(10, 10))
            sns.barplot(x='Importance', y='Feature', data=feature_importance_df, palette='viridis')
            plt.title(f'Feature Importance from {model_name}')
            plt.xlabel('Importance')
            plt.ylabel('Feature')
            plt.show()
        else:
            print(f"{model_name} does not have the feature_importances_ attribute.")

# Apply the function
plot_feature_importances(trained_models, X_train)
```

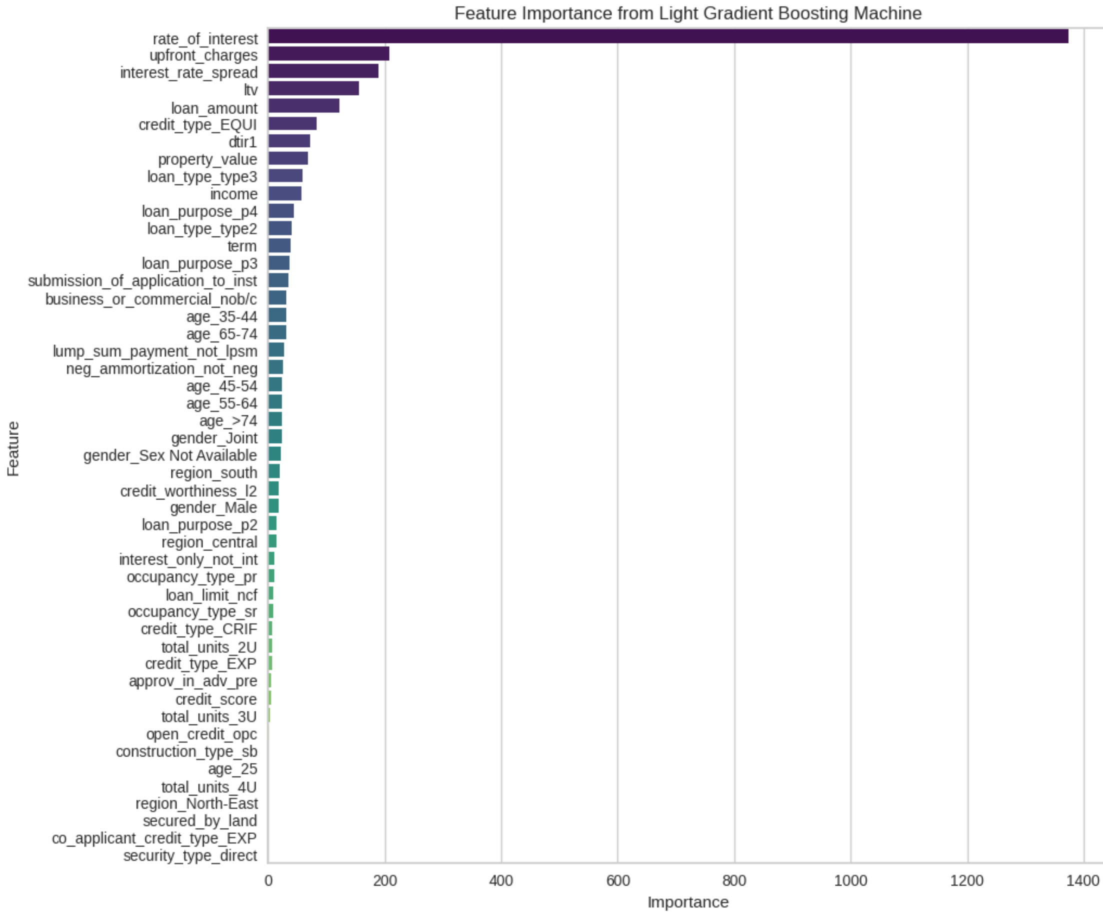




Feature Importance from Random Forest
credit_type_EQUI
rate_of_interest
upfront_charges
interest_rate_spread
ltv
dtir1
property_value
submission_of_application_to_inst
loan_amount
income
co_applicant_credit_type_EXP
loan_type_type2
credit_score
gender_Male
term
loan_purpose_p3
region_south
loan_purpose_p4
gender_Sex Not Available
gender_Joint
business_or_commercial_nob/c
age_55-64
age_45-54
age_65-74
age_35-44
loan_type_type3
credit_type_CRIF
credit_type_EXP
loan_limit_ncf
neg_ammortization_not_neg
region_central
credit_worthiness_l2
approv_in_adv_pre
age_>74
lump_sum_payment_not_lpsm
loan_purpose_p2
occupancy_type_pr
interest_only_not_int
total_units_2U
occupancy_type_sr
region_North-East
age_25
total_units_3U
total_units_4U
open_credit_opc
secured_by_land
security_type_direct
construction_type_sb
Feature
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
Importance


Feature Importance from XGBoost
credit_type_EQUI
submission_of_application_to_inst
business_or_commercial_nob/c
gender_Joint
region_south
gender_Male
credit_type_EXP
rate_of_interest
gender_Sex Not Available
loan_type_type2
loan_purpose_p3
upfront_charges
loan_limit_ncf
age_45-54
loan_purpose_p4
age_65-74
region_central
age_35-44
age_55-64
loan_purpose_p2
lump_sum_payment_not_lpsm
credit_type_CRIF
term
age_>74
loan_type_type3
neg_ammortization_not_neg
dtir1
loan_amount
interest_rate_spread
credit_worthiness_l2
ltv
interest_only_not_int
property_value
occupancy_type_sr
total_units_2U
income
total_units_3U
approv_in_adv_pre
occupancy_type_pr
open_credit_opc
secured_by_land
credit_score
co_applicant_credit_type_EXP
age_25
total_units_4U
construction_type_sb
region_North-East
security_type_direct
Feature
0.0
0.1
0.2
0.3
0.4
0.5
Importance


Feature Importance from AdaBoost
rate_of_interest
upfront_charges
age_55-64
loan_amount
loan_purpose_p4
credit_worthiness_l2
submission_of_application_to_inst
business_or_commercial_nob/c
neg_ammortization_not_neg
age_>74
lump_sum_payment_not_lpsm
credit_type_EXP
age_65-74
age_45-54
age_35-44
credit_type_CRIF
loan_purpose_p3
loan_purpose_p2
loan_type_type3
loan_type_type2
interest_rate_spread
term
property_value
income
region_south
ltv
dtir1
loan_limit_ncf
gender_Joint
gender_Male
region_central
credit_type_EQUI
age_25
co_applicant_credit_type_EXP
region_North-East
interest_only_not_int
total_units_4U
total_units_3U
total_units_2U
secured_by_land
occupancy_type_sr
occupancy_type_pr
construction_type_sb
open_credit_opc
approv_in_adv_pre
gender_Sex Not Available
credit_score
security_type_direct
Feature
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Importance



2.1.6 The best model is the xgboost as it incorporates the most features

2.2 Actionable Insights:

Insights into Factors Impacting Loan Default Risk

The analysis of the dataset reveals several key factors that significantly influence loan default risk:

2.2.1 Positively Correlated Variables

1. **Co-Applicant Credit Type (0.142)**: The correlation suggests that loans with co-applicants who have specific credit types (like 'CIB' or 'EXP') are more likely to default. This could imply that loans with certain co-applicant profiles might pose higher risks, potentially due to their credit history or other associated factors.
2. **Submission of Application (0.121)**: A positive correlation with loan defaults indicates that the method or timing of application submission ('to_inst' or 'not_inst') plays a role. Applications not submitted to institutions or submitted later may have higher default rates, possibly due to less stringent verification or urgency.
3. **Credit Type (0.112)**: The applicant's primary credit source (such as 'EXP', 'EQUI', etc.)

also shows a positive association with loan defaults. This could reflect varying levels of risk associated with different credit bureaus or their reporting standards.

4. **DTI Ratio (0.063)**: The Debt-to-Income ratio, though weakly correlated, still impacts the default risk. Higher DTI ratios indicate a greater likelihood of default, as borrowers with higher debt obligations relative to their income may struggle to repay.

2.2.2 Negatively Correlated Variables

1. **Income (-0.063)**: Higher income levels are negatively correlated with loan defaults, as borrowers with more substantial financial resources are generally more capable of repaying their loans.
2. **Business or Commercial Loans (-0.089)**: Loans categorized for business or commercial purposes are less likely to default, possibly due to the more stringent vetting processes or the presence of additional collateral.
3. **Negative Amortization (-0.134)**: Loans with negative amortization (where the payment is less than the interest, causing the loan balance to increase) show a negative correlation, suggesting these loans may be structured to mitigate default risks.
4. **Lump Sum Payment (-0.192)**: A strong negative correlation indicates that loans with lump sum payments have a lower risk of default. This might be because these payments often occur when borrowers have a sudden influx of capital, improving their repayment capacity.

To integrate the predictive model into loan approval processes, financial institutions can consider the following recommendations:

`1. Automated Pre-Screening System`

The model can be used to automate the initial loan pre-screening stage. By inputting applicant data (e.g., credit type, submission method, income level, debt-to-income ratio) into the model, the system can quickly assess the default risk for each applicant. This will allow institutions to efficiently identify high-risk applications, prioritizing those with a lower risk profile for further manual review. This approach enhances decision-making speed and accuracy, while also freeing up resources to focus on complex cases.

`2. Dynamic Risk-Based Pricing`

Financial institutions can employ the model to develop a dynamic risk-based pricing strategy. By evaluating an applicant's default risk, the model can help determine appropriate interest rates and loan terms that reflect the associated risk levels. Applicants with higher predicted default risks might be offered higher interest rates or shorter loan terms, while those with lower risks could benefit from more favourable terms. This strategy not only helps to balance risk and reward but also encourages responsible borrowing behaviour by aligning loan terms with risk profiles.

`3. Enhanced Risk Management Framework`

Incorporate the model's predictions into the institution's broader risk management framework. The model's outputs can be used alongside existing risk assessment tools to provide a more holistic view of potential default risks. This integration will allow institutions to adjust lending policies, set more precise credit limits, and make data-driven decisions about which loan products to promote or phase out. The model can also be used for ongoing monitoring, flagging loans that become riskier over time, allowing for timely intervention.

`4. Customized Credit Products`

Leverage the model's insights to develop customized credit products tailored to different risk profiles. For example, high-risk applicants could be offered loans with built-in safeguards, such as collateral requirements or co-signers, while low-risk applicants could receive incentives like lower rates or faster approvals. Tailoring products in this way ensures that the institution maximizes market reach while maintaining a balanced risk portfolio.

`5. Continuous Model Improvement and Validation`

The model should be regularly updated and validated with new data to ensure its accuracy and relevance over time. As market conditions, economic environments, and customer behaviours change, continuous model evaluation and recalibration are essential for maintaining robust risk predictions. Financial institutions should establish a feedback loop to incorporate outcomes from approved loans back into the model, refining it based on actual default occurrences.

By implementing these strategies, financial institutions can effectively integrate the predictive model into their loan approval processes, thereby enhancing decision-making, reducing default risks, and improving overall portfolio performance.