In [1]:

```
!pip install torch
!pip install torch-geometric
!pip install xgboost lightgbm catboost
!pip install streamlit
!pip install pyngrok
!pip install dask[dataframe]
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.1+cu
121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist
-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from
torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from to
rch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from to
rch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (
from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packa
ges (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages
(from jinja2->torch) (3.0.2)
Collecting torch-geometric
  Downloading torch_geometric-2.6.1-py3-none-any.whl.metadata (63 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.1/63.1 kB 2.5 MB/s eta 0:00:00
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from t
orch-geometric) (3.11.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from to
rch-geometric) (2024.10.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from to
rch-geometric) (3.1.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from tor
ch-geometric) (1.26.4)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-packages (
from torch-geometric) (5.9.5)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-packages (from
torch-geometric) (3.2.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from
torch-geometric) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from torc
h-geometric) (4.66.6)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-package
s (from aiohttp->torch-geometric) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (
from aiohttp->torch-geometric) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packag
es (from aiohttp->torch-geometric) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-pack
ages (from aiohttp->torch-geometric) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-package
s (from aiohttp->torch-geometric) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packag
es (from aiohttp->torch-geometric) (1.17.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (4.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages
(from jinja2->torch-geometric) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist
-packages (from requests->torch-geometric) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (f
rom requests->torch-geometric) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packa
```

```
ges (from requests->torch-geometric) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packa
ges (from requests->torch-geometric) (2024.8.30)
Requirement already satisfied: typing-extensions>=4.1.0 in /usr/local/lib/python3.10/dist
-packages (from multidict<7.0,>=4.5->aiohttp->torch-geometric) (4.12.2)
Downloading torch_geometric-2.6.1-py3-none-any.whl (1.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 17.4 MB/s eta 0:00:00
Installing collected packages: torch-geometric
Successfully installed torch-geometric-2.6.1
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.1.2)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-packages (4.5.0
)
Collecting catboost
  Downloading catboost-1.2.7-cp310-cp310-manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgb
oost) (1.26.4)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.10/dist-package
s (from xgboost) (2.23.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgb
oost) (1.13.1)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from
catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (fro
m catboost) (3.8.0)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (f
rom catboost) (2.2.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from ca
tboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catbo
ost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-p
ackages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (f
rom pandas>=0.24->catboost) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages
(from pandas>=0.24->catboost) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-package
s (from matplotlib->catboost) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (f
rom matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib->catboost) (4.55.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib->catboost) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages
(from matplotlib->catboost) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (
from matplotlib->catboost) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-package
s (from matplotlib->catboost) (3.2.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages
(from plotly->catboost) (9.0.0)
Downloading catboost-1.2.7-cp310-cp310-manylinux2014_x86_64.whl (98.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.7/98.7 MB 6.5 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.7
Collecting streamlit
  Downloading streamlit-1.40.2-py2.py3-none-any.whl.metadata (8.4 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages
(from streamlit) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/local/lib/python3.10/dist-packag
es (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packa
ges (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (
from streamlit) (8.1.7)
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.10/dist-packages
(from streamlit) (1.26.4)
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.10/dist-packag
es (from streamlit) (24.2)
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.10/dist-package
s (from streamlit) (2.2.2)
```

```
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.10/dist-packag
es (from streamlit) (11.0.0)
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.10/dist-packag
es (from streamlit) (4.25.5)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-packages (f
rom streamlit) (17.0.0)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-packag
es (from streamlit) (2.32.3)
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-packag
es (from streamlit) (13.9.4)
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.10/dist-pack
ages (from streamlit) (9.0.0)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-packages
(from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.10/d
ist-packages (from streamlit) (4.12.2)
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 2.6 MB/s eta 0:00:00
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.10/
dist-packages (from streamlit) (3.1.43)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.10/dist-packag
es (from streamlit) (6.3.3)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (fr
om altair<6,>=4.0->streamlit) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from al
tair<6,>=4.0->streamlit) (3.1.4)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages
(from altair<6,>=4.0->streamlit) (4.23.0)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from alt
air<6,>=4.0->streamlit) (0.12.1)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-packages
(from gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.11)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-p
ackages (from pandas<3,>=1.4.0->streamlit) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (f
rom pandas<3,>=1.4.0->streamlit) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages
(from pandas<3,>=1.4.0->streamlit) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist
-packages (from requests<3,>=2.27->streamlit) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (f
rom requests<3,>=2.27->streamlit) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packa
ges (from requests<3,>=2.27->streamlit) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packa
ges (from requests<3,>=2.27->streamlit) (2024.8.30)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-pa
ckages (from rich<14,>=10.14.0->streamlit) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-
packages (from rich<14,>=10.14.0->streamlit) (2.18.0)
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages
(from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->streamlit) (5.0.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages
(from jinja2->altair<6,>=4.0->streamlit) (3.0.2)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (
from jsonschema>=3.0->altair<6,>=4.0->streamlit) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/pyt
hon3.10/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-pack
ages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages
(from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.21.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (fro
m markdown-it-py>=2.2.0->rich<14,>=10.14.0->streamlit) (0.1.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit) (1.16.0)
Downloading streamlit-1.40.2-py2.py3-none-any.whl (8.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.6/8.6 MB 19.0 MB/s eta 0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 86.0 MB/s eta 0:00:00
Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (79 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 4.9 MB/s eta 0:00:00
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.40.2 watchdog-6.0.0
Collecting pyngrok
  Downloading pyngrok-7.2.1-py3-none-any.whl.metadata (8.3 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.10/dist-packages (fr
om pyngrok) (6.0.2)
Downloading pyngrok-7.2.1-py3-none-any.whl (22 kB)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.2.1
Requirement already satisfied: dask[dataframe] in /usr/local/lib/python3.10/dist-packages
(2024.10.0)
Requirement already satisfied: click>=8.1 in /usr/local/lib/python3.10/dist-packages (fro
m dask[dataframe]) (8.1.7)
Requirement already satisfied: cloudpickle>=3.0.0 in /usr/local/lib/python3.10/dist-packa
ges (from dask[dataframe]) (3.1.0)
Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.10/dist-packag
es (from dask[dataframe]) (2024.10.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages
(from dask[dataframe]) (24.2)
Requirement already satisfied: partd>=1.4.0 in /usr/local/lib/python3.10/dist-packages (f
rom dask[dataframe]) (1.4.2)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (
from dask[dataframe]) (6.0.2)
Requirement already satisfied: toolz>=0.10.0 in /usr/local/lib/python3.10/dist-packages (
from dask[dataframe]) (0.12.1)
Requirement already satisfied: importlib-metadata>=4.13.0 in /usr/local/lib/python3.10/di
st-packages (from dask[dataframe]) (8.5.0)
Requirement already satisfied: pandas>=2.0 in /usr/local/lib/python3.10/dist-packages (fr
om dask[dataframe]) (2.2.2)
Collecting dask-expr<1.2,>=1.1 (from dask[dataframe])
  Downloading dask_expr-1.1.19-py3-none-any.whl.metadata (2.6 kB)
INFO: pip is looking at multiple versions of dask-expr to determine which version is comp
atible with other requirements. This could take a while.
  Downloading dask_expr-1.1.18-py3-none-any.whl.metadata (2.6 kB)
  Downloading dask_expr-1.1.16-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: pyarrow>=14.0.1 in /usr/local/lib/python3.10/dist-packages
(from dask-expr<1.2,>=1.1->dask[dataframe]) (17.0.0)
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.10/dist-packages (fro
m importlib-metadata>=4.13.0->dask[dataframe]) (3.21.0)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (
from pandas>=2.0->dask[dataframe]) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-p
ackages (from pandas>=2.0->dask[dataframe]) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (f
rom pandas>=2.0->dask[dataframe]) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages
(from pandas>=2.0->dask[dataframe]) (2024.2)
Requirement already satisfied: locket in /usr/local/lib/python3.10/dist-packages (from pa
rtd>=1.4.0->dask[dataframe]) (1.0.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
python-dateutil>=2.8.2->pandas>=2.0->dask[dataframe]) (1.16.0)
Downloading dask_expr-1.1.16-py3-none-any.whl (243 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 243.2/243.2 kB 6.0 MB/s eta 0:00:00
Installing collected packages: dask-expr
Successfully installed dask-expr-1.1.16
```

Imported necessary libraries

In [2]:

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
```

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from collections import Counter
from imblearn.over_sampling import SMOTE
import xgboost as xgb
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import make_scorer
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from torch_geometric.data import Data
from catboost import CatBoostClassifier
import xgboost as xgb
import lightgbm as lgb
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
import joblib
import streamlit as st
from pyngrok import ngrok
```

Loading Loan Approval Prediction Dataset

In [3]:

```python
# Loading the data
train_data = pd.read_excel('/content/Loan_TrainData.xlsx')
train_data.head(5)
```

Out[3]:

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coapplicant_Income	Loan_Amount	Term	Cre
0	Male	No	0.0	Graduate	No	584900	0.0	15000000	360.0	
1	Male	Yes	1.0	Graduate	No	458300	150800.0	12800000	360.0	
2	Male	Yes	0.0	Graduate	Yes	300000	0.0	6600000	360.0	
3	Male	Yes	0.0	Not Graduate	No	258300	235800.0	12000000	360.0	
4	Male	No	0.0	Graduate	No	600000	0.0	14100000	360.0	

In [4]:

```python
# Display summary statistics for numerical columns
numerical_summary = train_data.describe()

print("Numerical Summary of Loan Dataset:\n", numerical_summary)
```

```
Numerical Summary of Loan Dataset:
        Dependents  Applicant_Income  Coapplicant_Income   Loan_Amount  \
count  599.000000      6.140000e+02        6.140000e+02  6.140000e+02
mean     0.762938      5.403459e+05        1.621246e+05  1.414104e+07
std      1.015216      6.109042e+05        2.926248e+05  8.815682e+06
min      0.000000      1.500000e+04        0.000000e+00  0.000000e+00
25%      0.000000      2.877500e+05        0.000000e+00  9.800000e+06
50%      0.000000      3.812500e+05        1.188500e+05  1.250000e+07
75%      2.000000      5.795000e+05        2.297250e+05  1.647500e+07
max      3.000000      8.100000e+06        4.166700e+06  7.000000e+07

            Term  Credit_History
count  600.00000      564.000000
mean   342.00000        0.842199
std     65.12041        0.364878
min     12.00000        0.000000
25%    360.00000        1.000000
50%    360.00000        1.000000
75%    360.00000        1.000000
```

```
max     480.00000         1.000000
```

In [5]:

```
#categorical columns summary
categorical_summary = train_data.describe(include=['object'])
print("\nCategorical Summary of Loan Dataset:\n", categorical_summary)
```

```
Categorical Summary of Loan Dataset:
        Gender Married Education Self_Employed       Area Status
count     601     611       614           582        614    614
unique      2       2         2             2          3      2
top      Male     Yes  Graduate            No  Semiurban      Y
freq      489     398       480           500        233    422
```

In [6]:

```
# Checking  for missing values in each column
missing_values = train_data.isnull().sum()
print("\nMissing Values in Each Column:\n", missing_values)
```

```
Missing Values in Each Column:
 Gender                13
Married                3
Dependents            15
Education              0
Self_Employed         32
Applicant_Income       0
Coapplicant_Income     0
Loan_Amount            0
Term                  14
Credit_History        50
Area                   0
Status                 0
dtype: int64
```

In [7]:

```
for column in train_data.columns:
    if column in ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Cred
it_History']:
        train_data[column] = train_data[column].fillna(train_data[column].mode()[0])
    elif train_data[column].dtype in ['float64', 'int64']:
        train_data[column] = train_data[column].fillna(train_data[column].mean())

# Checking if there are any remaining missing values
missing_values_after_filling = train_data.isnull().sum()
print("Missing values after filling:")
print(missing_values_after_filling)
```

```
Missing values after filling:
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
Applicant_Income      0
Coapplicant_Income    0
Loan_Amount           0
Term                  0
Credit_History        0
Area                  0
Status                0
dtype: int64
```

In [8]:

```
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for column in train_data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    train_data[column] = le.fit_transform(train_data[column])
```

```
    label_encoders[column] = le
train_data.head(5)
```

Out[8]:

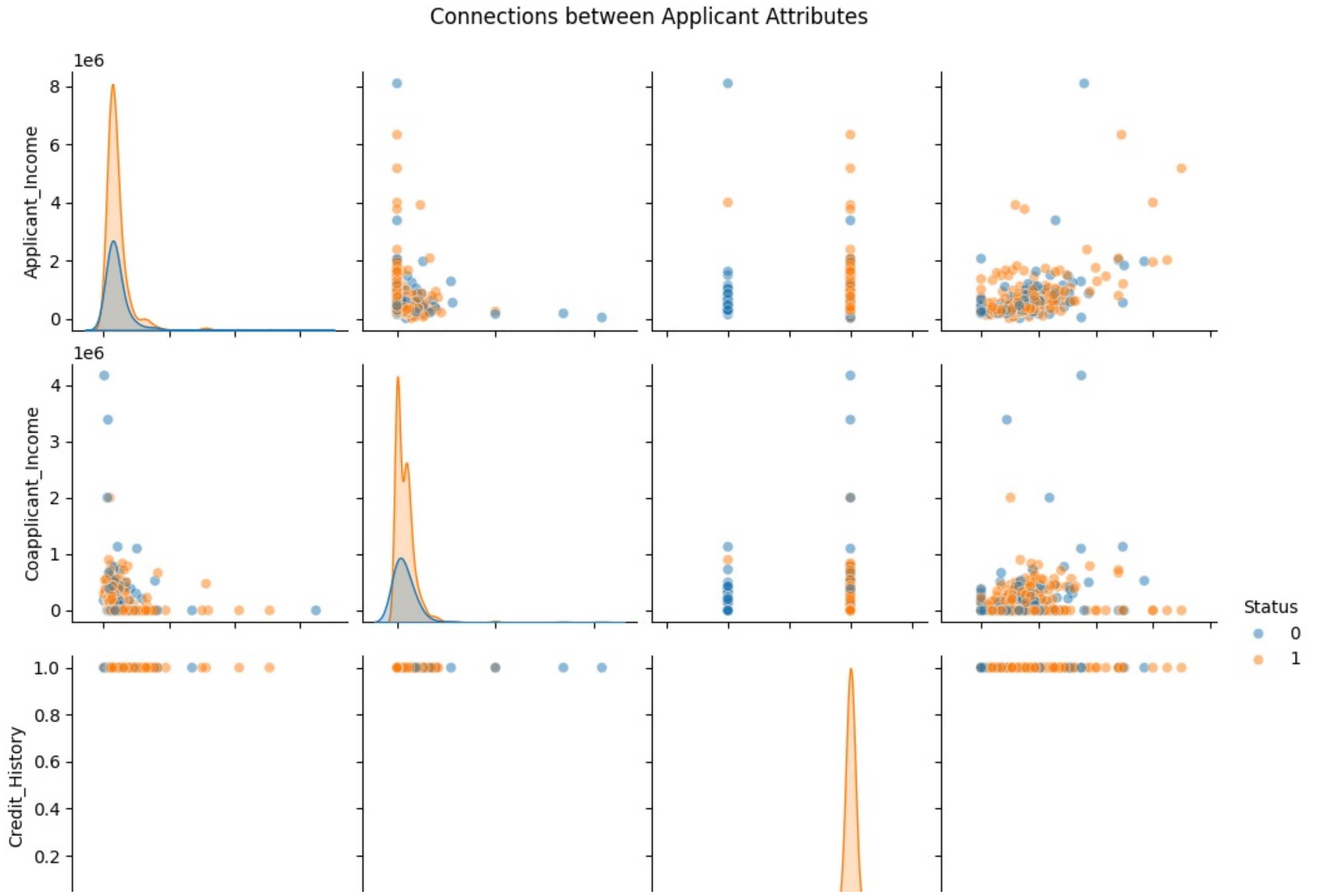
	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coapplicant_Income	Loan_Amount	Term	Cre
0	1	0	0.0	0	0	584900	0.0	15000000	360.0	
1	1	1	1.0	0	0	458300	150800.0	12800000	360.0	
2	1	1	0.0	0	1	300000	0.0	6600000	360.0	
3	1	1	0.0	1	0	258300	235800.0	12000000	360.0	
4	1	0	0.0	0	0	600000	0.0	14100000	360.0	

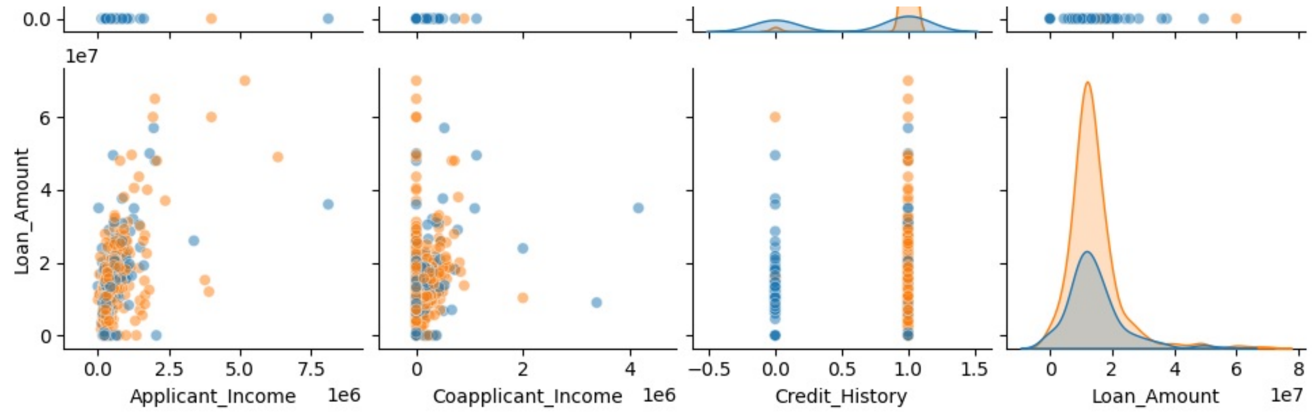
Feature Relationship Analysis

In [9]:

```
selected_columns = ["Applicant_Income", "Coapplicant_Income", "Credit_History", "Loan_Amo
unt", "Status"]

selected_data = train_data[selected_columns]
selected_data['Status'] = selected_data['Status'].astype('category')

#A pairplot to visualize pairwise relationships between the selected features
sns.pairplot(selected_data, hue="Status", plot_kws={'alpha': 0.5})
plt.suptitle("Connections between Applicant Attributes", y=1.02)
plt.show()
```

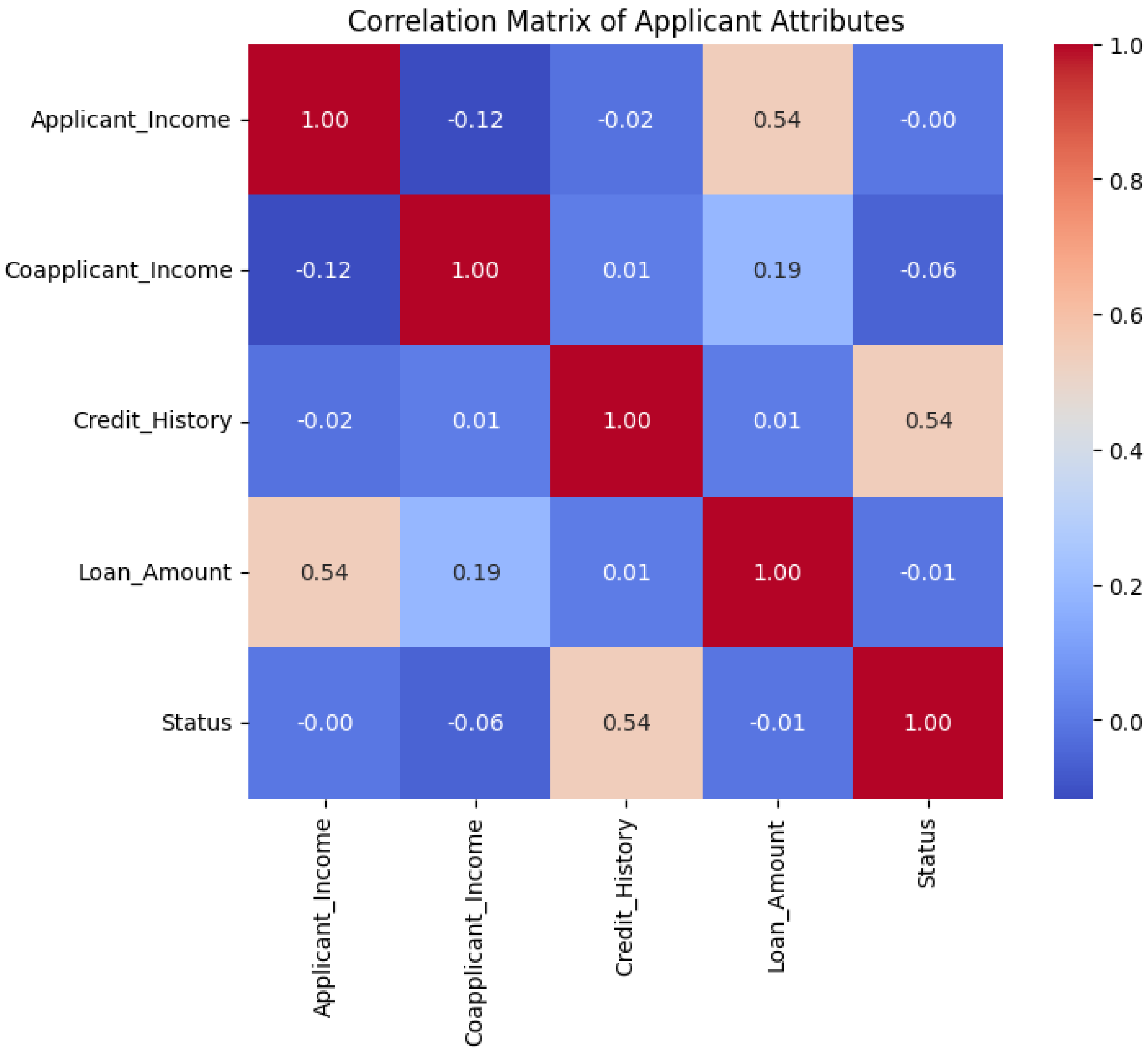
```
<ipython-input-9-cc7be0762fde>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  selected_data['Status'] = selected_data['Status'].astype('category')
```





In [10]:

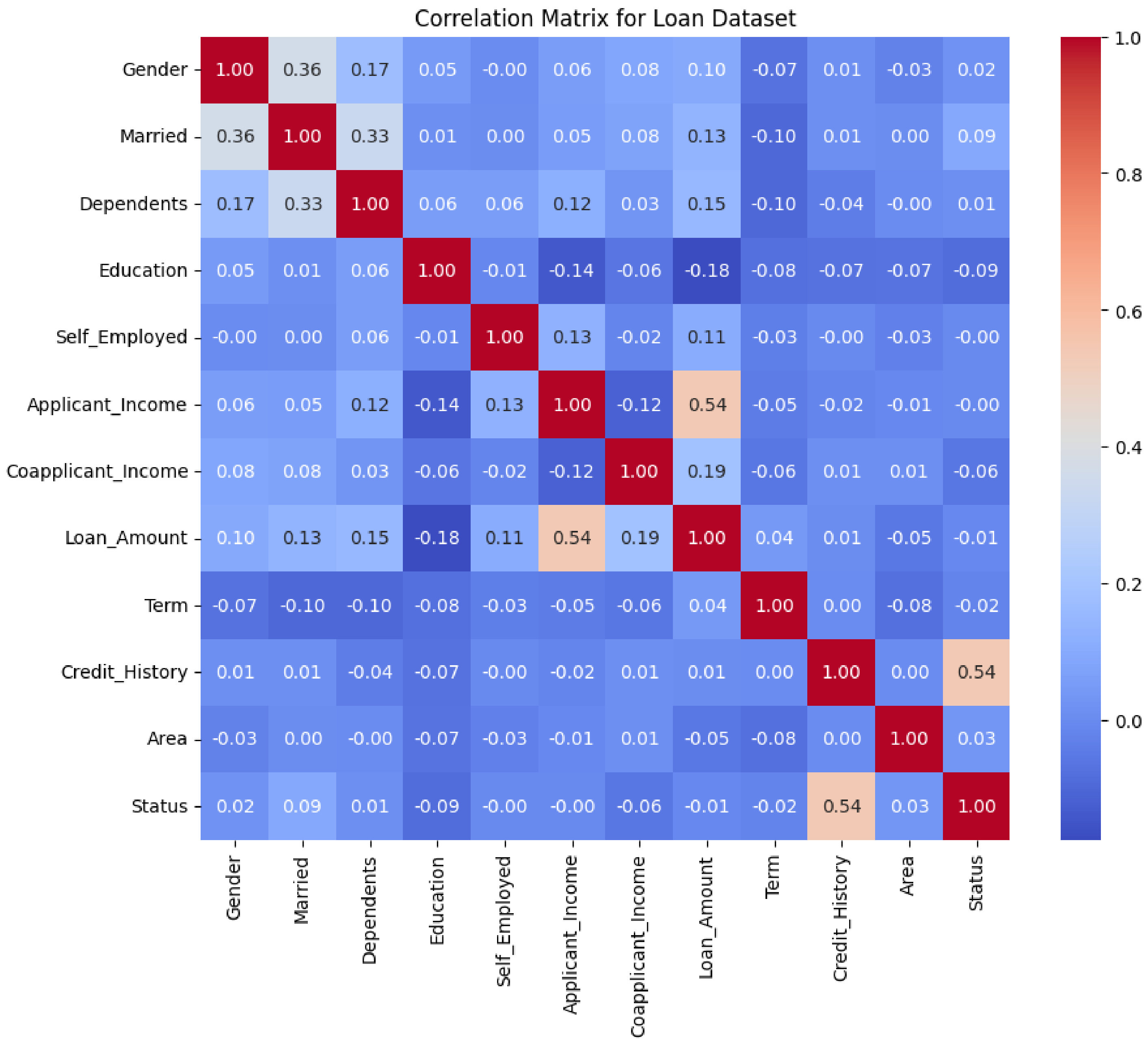
```
plt.figure(figsize=(8, 6))
correlation_matrix = selected_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", square=True)
plt.title("Correlation Matrix of Applicant Attributes")
plt.show()
```



Correlation Matrix Heatmap

In [11]:

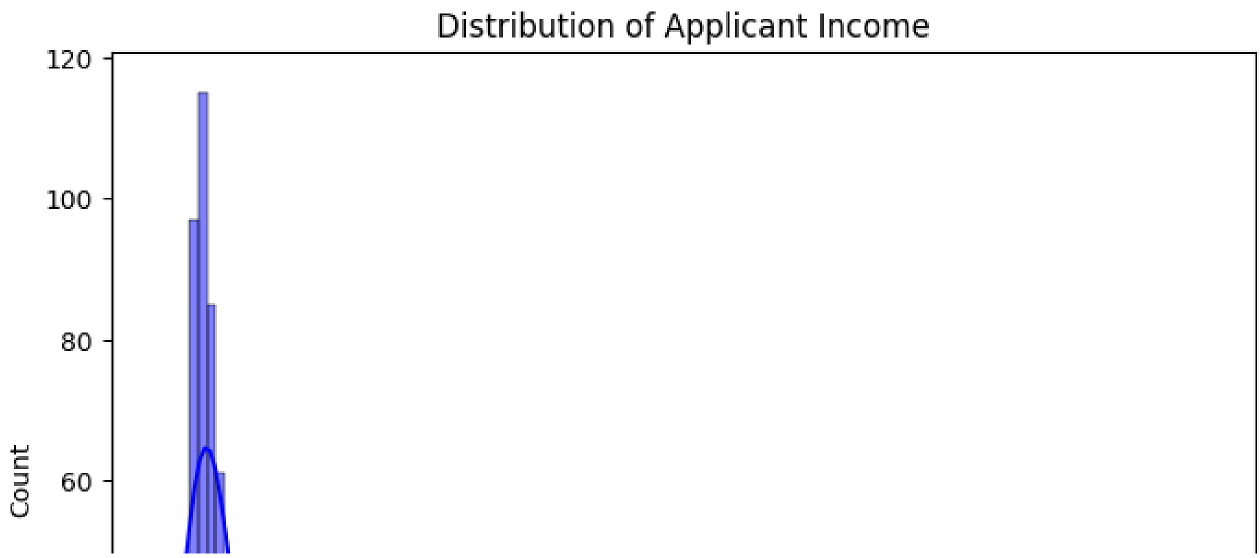
```
plt.figure(figsize=(10, 8))
sns.heatmap(train_data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix for Loan Dataset')
plt.show()
```

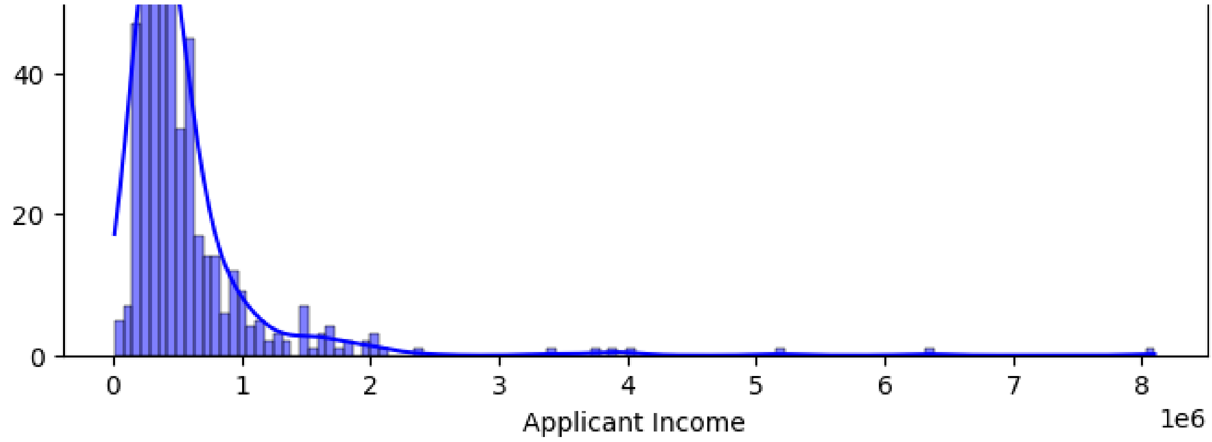


Distribution of Applicant Income

In [12]:

```
plt.figure(figsize=(8, 6))
sns.histplot(train_data['Applicant_Income'], kde=True, color='blue')
plt.title('Distribution of Applicant Income')
plt.xlabel('Applicant Income')
plt.show()
```

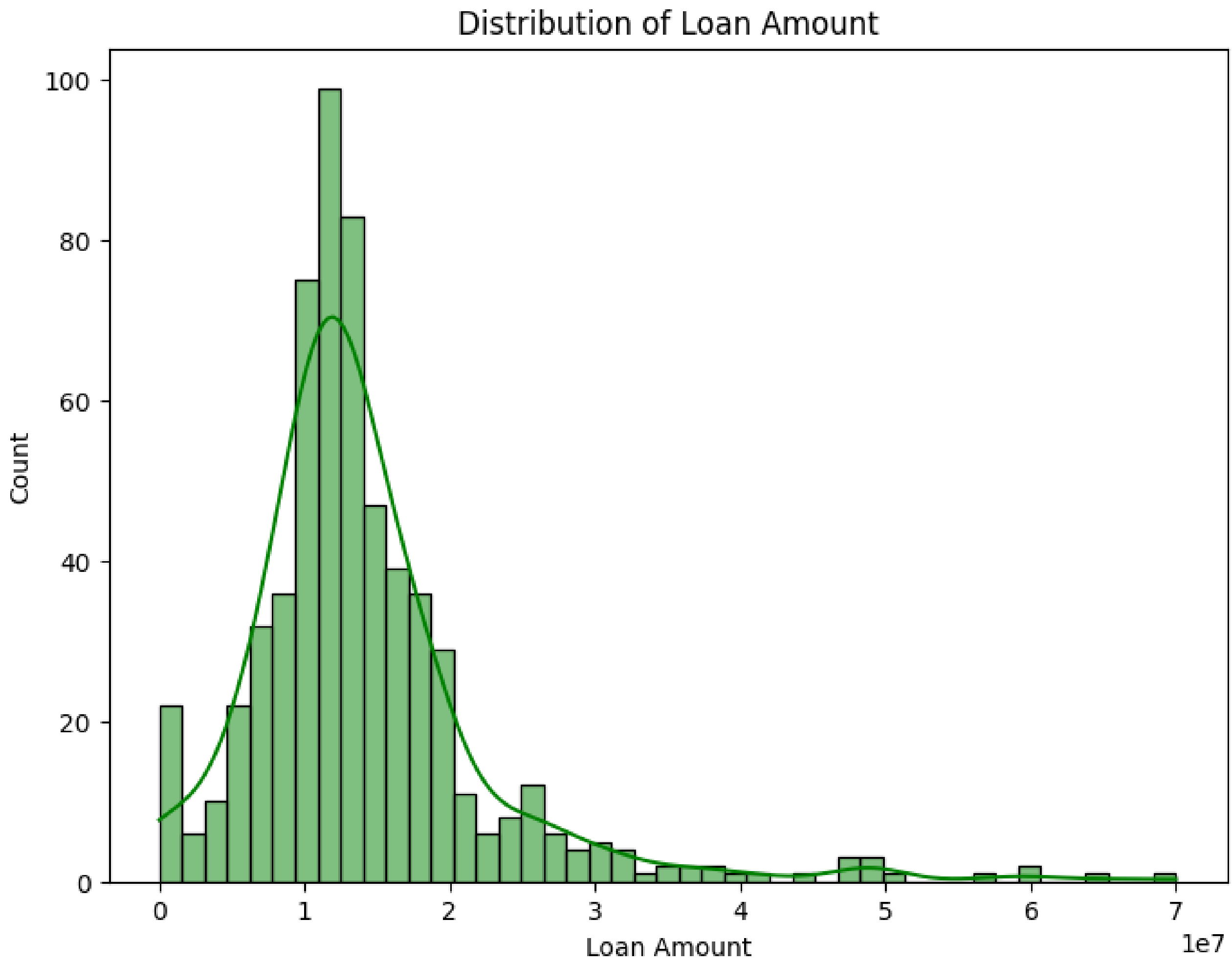




Distribution of Loan Amount

In [13]:

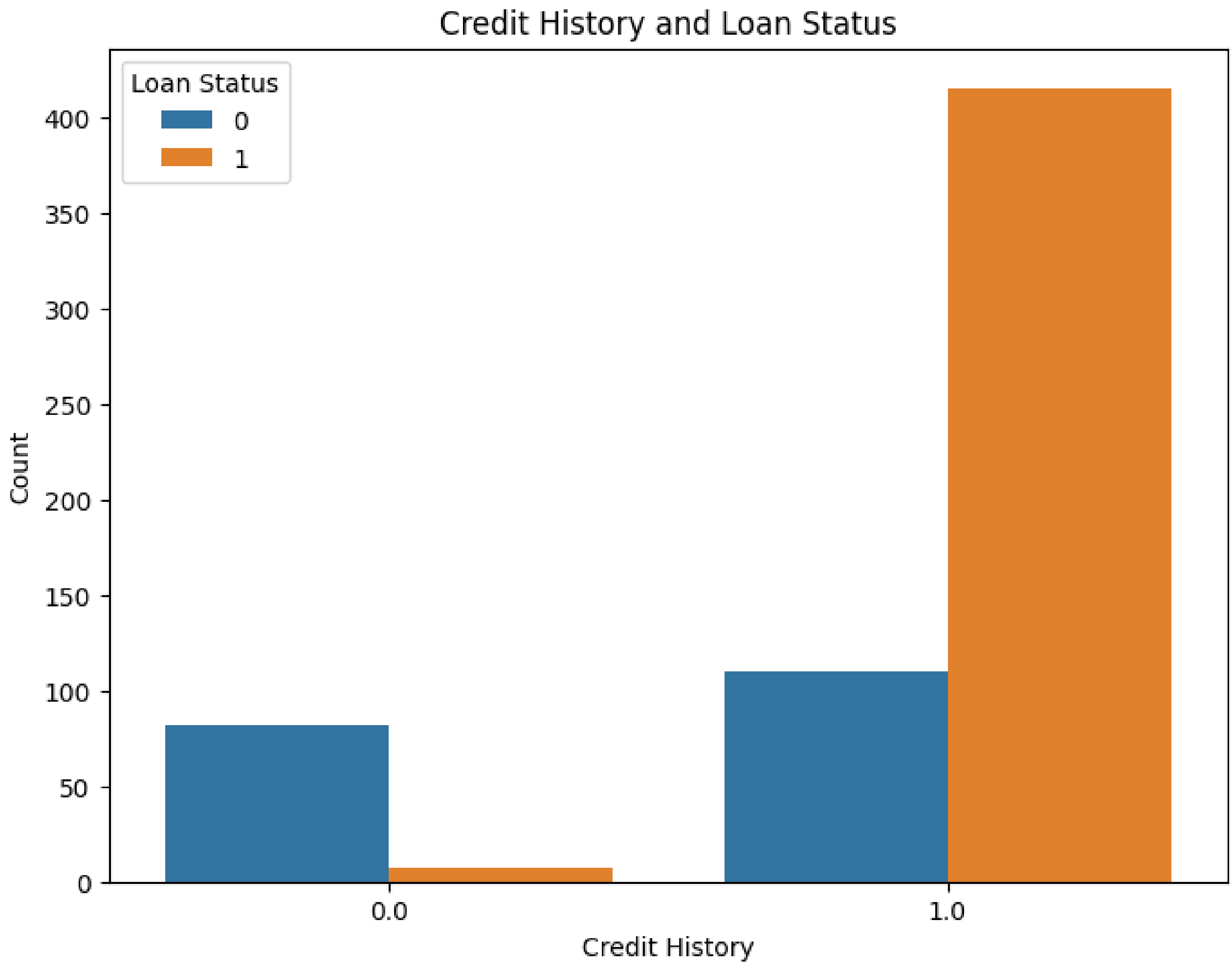
```
plt.figure(figsize=(8, 6))
sns.histplot(train_data['Loan_Amount'], kde=True, color='green')
plt.title('Distribution of Loan Amount')
plt.xlabel('Loan Amount')
plt.show()
```



Credit History Distribution by Loan Status

In [14]:

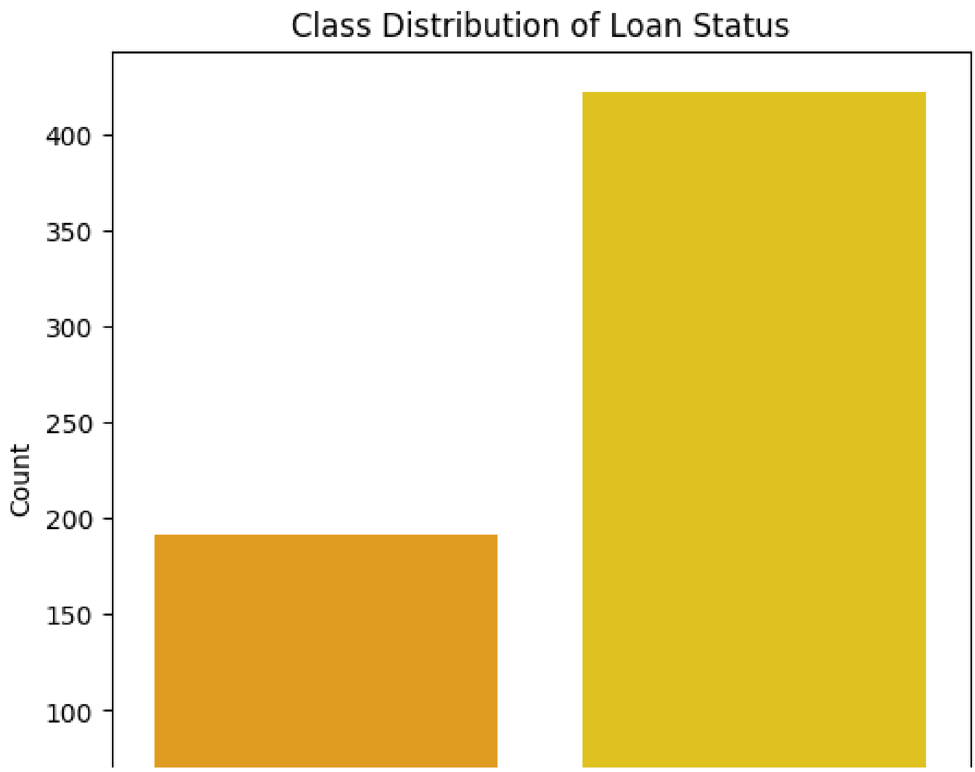
```
plt.figure(figsize=(8, 6))
sns.countplot(data=train_data, x='Credit_History', hue='Status')
plt.title('Credit History and Loan Status')
plt.xlabel('Credit History')
plt.ylabel('Count')
plt.legend(title='Loan Status')
plt.show()
```

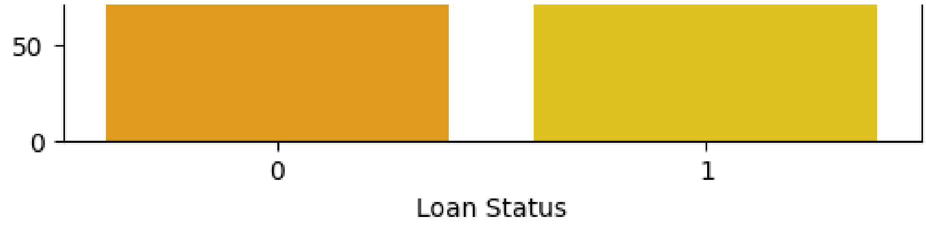


Target Variable Imbalance - Loan Status Distribution

In [15]:

```
plt.figure(figsize=(6, 6))
sns.countplot(data=train_data, x='Status', hue='Status', palette=['#FFA500', '#FFD700'],
legend=False)
plt.title('Class Distribution of Loan Status')
plt.xlabel('Loan Status')
plt.ylabel('Count')
plt.show()
```





In [16]:

```
# Feature Engineering and scaling numerical features
train_data['Total_Income'] = train_data['Applicant_Income'] + train_data['Coapplicant_Inc
ome']
train_data['Loan_Income_Ratio'] = train_data['Loan_Amount'] / train_data['Total_Income']
train_data['Income_Per_Dependent'] = train_data['Applicant_Income'] / (train_data['Depen
dents'] + 1)

scaler = StandardScaler()
numerical_columns = ['Applicant_Income', 'Coapplicant_Income', 'Loan_Amount', 'Total_Inco
me', 'Loan_Income_Ratio', 'Income_Per_Dependent']
train_data[numerical_columns] = scaler.fit_transform(train_data[numerical_columns])

train_data.to_csv("processed_data.csv", index=False)

X = train_data.drop('Status', axis=1)
y = train_data['Status']
```

In [17]:

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

XGBoost Model Implementation

In [18]:

```
xgb_model = XGBClassifier(eval_metric='logloss')  # Removed use_label_encoder
xgb_model.fit(X_train, y_train)
y_pred_val = xgb_model.predict(X_val)
y_pred_proba_val = xgb_model.predict_proba(X_val)[:, 1]
metrics = {
    'accuracy': accuracy_score(y_val, y_pred_val),
    'precision': precision_score(y_val, y_pred_val),
    'recall': recall_score(y_val, y_pred_val),
    'f1_score': f1_score(y_val, y_pred_val),
    'roc_auc': roc_auc_score(y_val, y_pred_proba_val)
}

print("Performance metrics:", metrics)
```

```
Performance metrics: {'accuracy': 0.7642276422764228, 'precision': 0.7741935483870968, 'r
ecall': 0.9, 'f1_score': 0.8323699421965318, 'roc_auc': 0.7540697674418605}
```

In [19]:

```
#Defining initial XGBoost model with imbalance handling
xgb_model = XGBClassifier(
    eval_metric='logloss',
    scale_pos_weight=len(y_train[y_train == 0]) / len(y_train[y_train == 1])  # Imbalanc
e handling
)

param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'n_estimators': [100, 200, 300],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.3, 0.5, 0.8]
}
```

```
cv = StratifiedKFold(n_splits=5)
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=cv, scoring='r
oc_auc', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best model from grid search
best_xgb_model = grid_search.best_estimator_
y_pred_val = best_xgb_model.predict(X_val)
y_pred_proba_val = best_xgb_model.predict_proba(X_val)[:, 1]

metrics = {
    'accuracy': accuracy_score(y_val, y_pred_val),
    'precision': precision_score(y_val, y_pred_val),
    'recall': recall_score(y_val, y_pred_val),
    'f1_score': f1_score(y_val, y_pred_val),
    'roc_auc': roc_auc_score(y_val, y_pred_proba_val)
}

print("Performance metrics:", metrics)
print("Best parameters:", grid_search.best_params_)
```

```
Performance metrics: {'accuracy': 0.7642276422764228, 'precision': 0.8, 'recall': 0.85, '
f1_score': 0.8242424242424242, 'roc_auc': 0.7526162790697675}
Best parameters: {'colsample_bytree': 0.3, 'learning_rate': 0.1, 'max_depth': 5, 'n_estim
ators': 100, 'subsample': 1.0}
```

In [20]:

```
xgb_model = XGBClassifier(
    eval_metric='logloss',
    scale_pos_weight=len(y_train[y_train == 0]) / len(y_train[y_train == 1])
)

# Simplified parameter grid for RandomizedSearch
param_dist = {
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': np.arange(3, 6),
    'n_estimators': np.arange(100, 301, 50),
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.3, 0.5, 0.7],
    'gamma': [0, 0.1, 0.5]
}

cv = StratifiedKFold(n_splits=5)
random_search = RandomizedSearchCV(
    estimator=xgb_model,
    param_distributions=param_dist,
    n_iter=20,
    cv=cv,
    scoring='roc_auc',
    n_jobs=-1,
    random_state=42
)

random_search.fit(X_train, y_train)
best_xgb_model = random_search.best_estimator_
y_pred_val = best_xgb_model.predict(X_val)
y_pred_proba_val = best_xgb_model.predict_proba(X_val)[:, 1]

metrics = {
    'accuracy': accuracy_score(y_val, y_pred_val),
    'precision': precision_score(y_val, y_pred_val),
    'recall': recall_score(y_val, y_pred_val),
    'f1_score': f1_score(y_val, y_pred_val),
    'roc_auc': roc_auc_score(y_val, y_pred_proba_val)
}

print("Performance metrics:", metrics)
print("Best parameters:", random_search.best_params_)
```

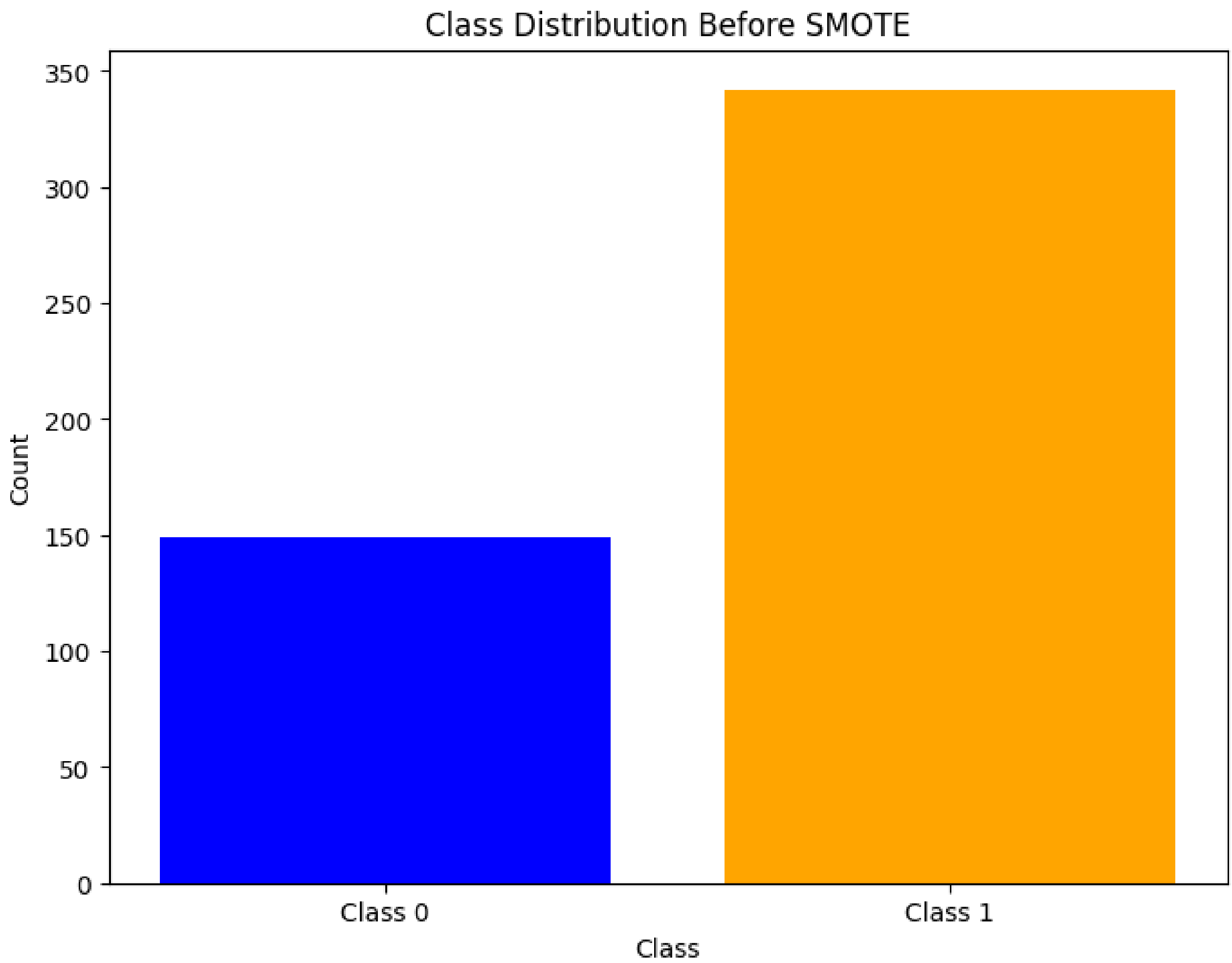
```
Performance metrics: {'accuracy': 0.7560975609756098, 'precision': 0.7840909090909091, 'r
ecall': 0.8625, 'f1_score': 0.8214285714285714, 'roc_auc': 0.7741279069767442}
```

```
Best parameters: {'subsample': 0.8, 'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.05, 'gamma': 0.5, 'colsample_bytree': 0.5}
```

In [21]:

```python
print("Class distribution before SMOTE:", Counter(y_train))
class_counts_before = Counter(y_train)

# Plot class distribution before SMOTE
plt.figure(figsize=(8, 6))
plt.bar(class_counts_before.keys(), class_counts_before.values(), color=['blue', 'orange'
])
plt.title("Class Distribution Before SMOTE")
plt.xlabel("Class")
plt.ylabel("Count")
plt.xticks([0, 1], ["Class 0", "Class 1"])
plt.show()
```

```
Class distribution before SMOTE: Counter({1: 342, 0: 149})
```



In [22]:

```python
smote = SMOTE(random_state=42)

X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
print("Class distribution after SMOTE:", Counter(y_train_resampled))
class_counts_after = Counter(y_train_resampled)

# Plot class distribution after SMOTE
plt.figure(figsize=(8, 6))
plt.bar(class_counts_after.keys(), class_counts_after.values(), color=['blue', 'orange'])
plt.title("Class Distribution After SMOTE")
plt.xlabel("Class")
plt.ylabel("Count")
plt.xticks([0, 1], ["Class 0", "Class 1"])
plt.show()
```

```
Class distribution after SMOTE: Counter({0: 342, 1: 342})
```

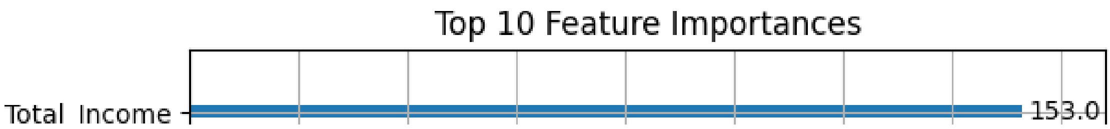
In [23]:

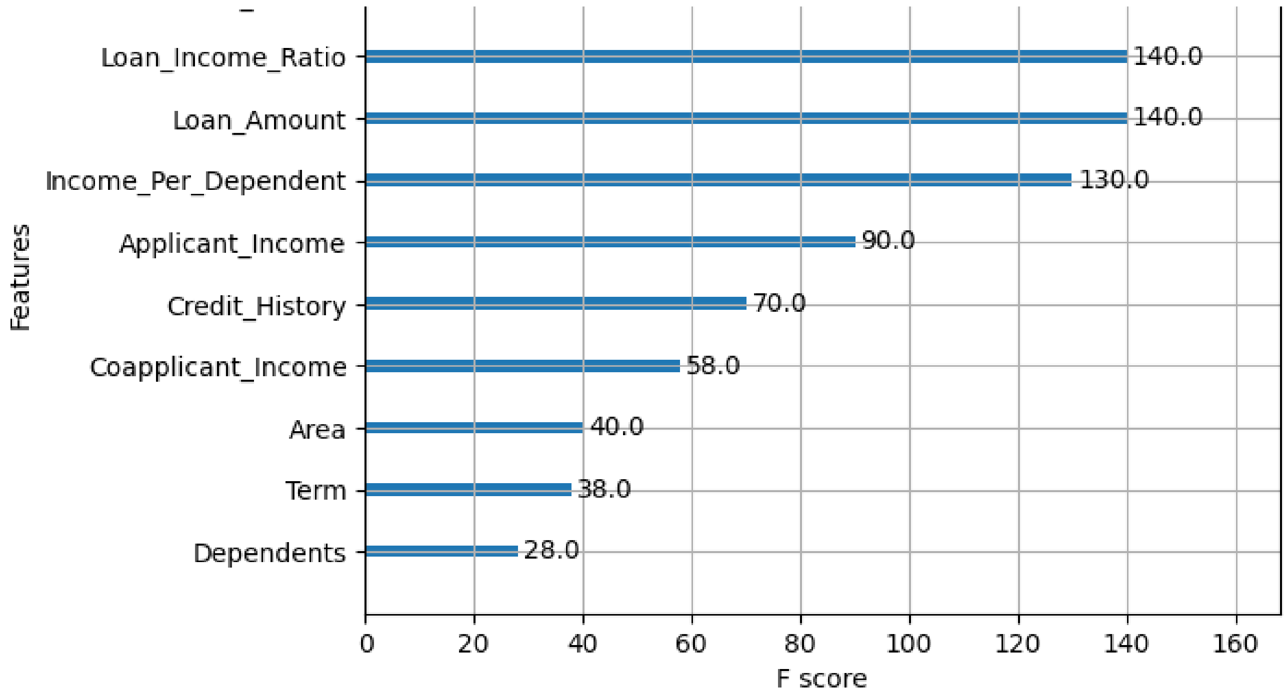
```
xgb_model = XGBClassifier(
    learning_rate=0.1,
    max_depth=5,
    n_estimators=200,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)

xgb_model.fit(X_train_resampled, y_train_resampled)
y_pred_val = xgb_model.predict(X_val)
y_pred_proba_val = xgb_model.predict_proba(X_val)[:, 1]
metrics = {
    'accuracy': accuracy_score(y_val, y_pred_val),
    'precision': precision_score(y_val, y_pred_val),
    'recall': recall_score(y_val, y_pred_val),
    'f1_score': f1_score(y_val, y_pred_val),
    'roc_auc': roc_auc_score(y_val, y_pred_proba_val)
}

print("Performance metrics:", metrics)
```

```
Performance metrics: {'accuracy': 0.7642276422764228, 'precision': 0.7865168539325843, 'r
ecall': 0.875, 'f1_score': 0.8284023668639053, 'roc_auc': 0.7479651162790697}
```

In [24]:

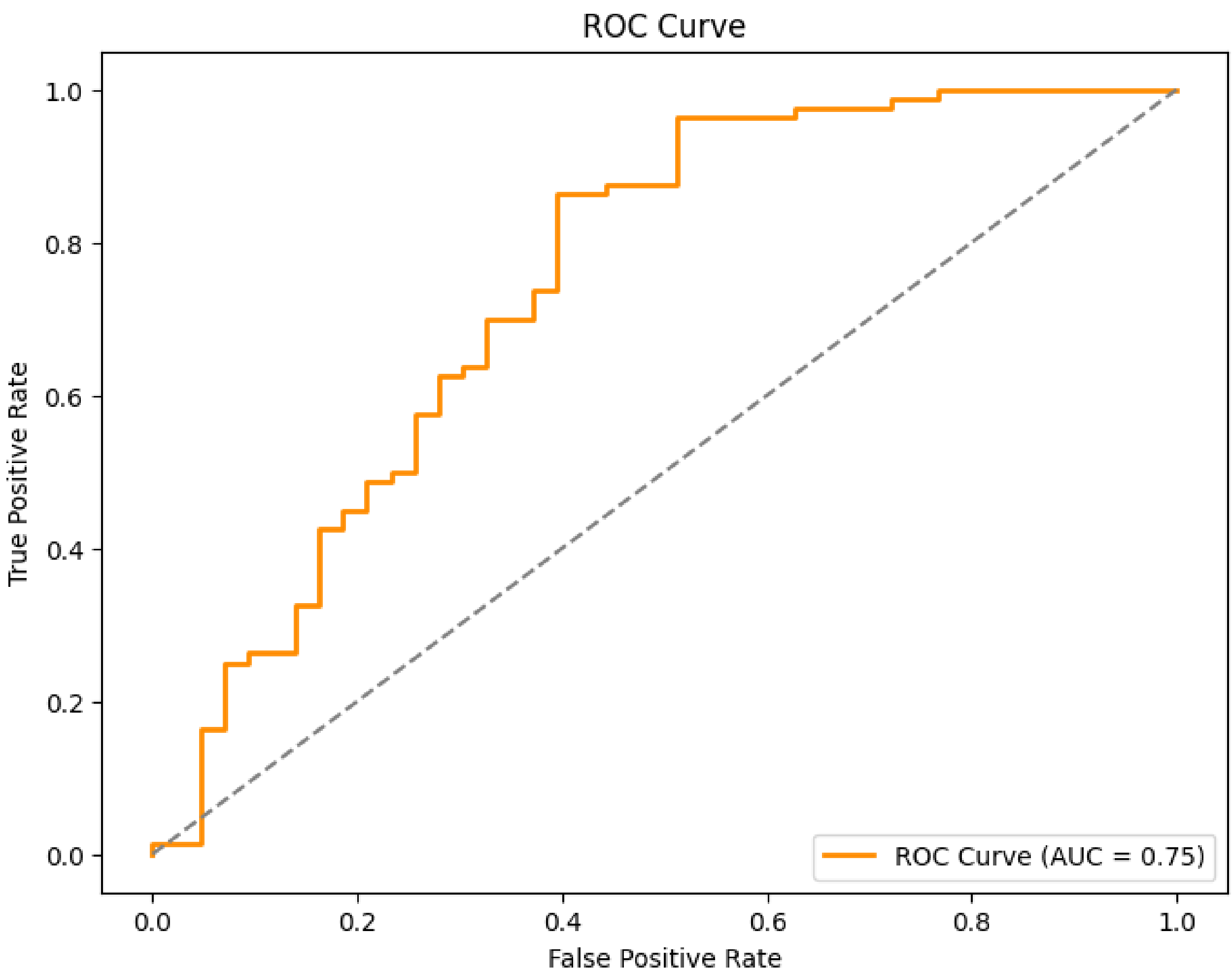
```
# Plot feature importance
xgb.plot_importance(best_xgb_model, importance_type='weight', max_num_features=10)
plt.title('Top 10 Feature Importances')
plt.show()
```





In [25]:

```
fpr, tpr, thresholds = roc_curve(y_val, y_pred_proba_val)
roc_auc = auc(fpr, tpr)
# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```



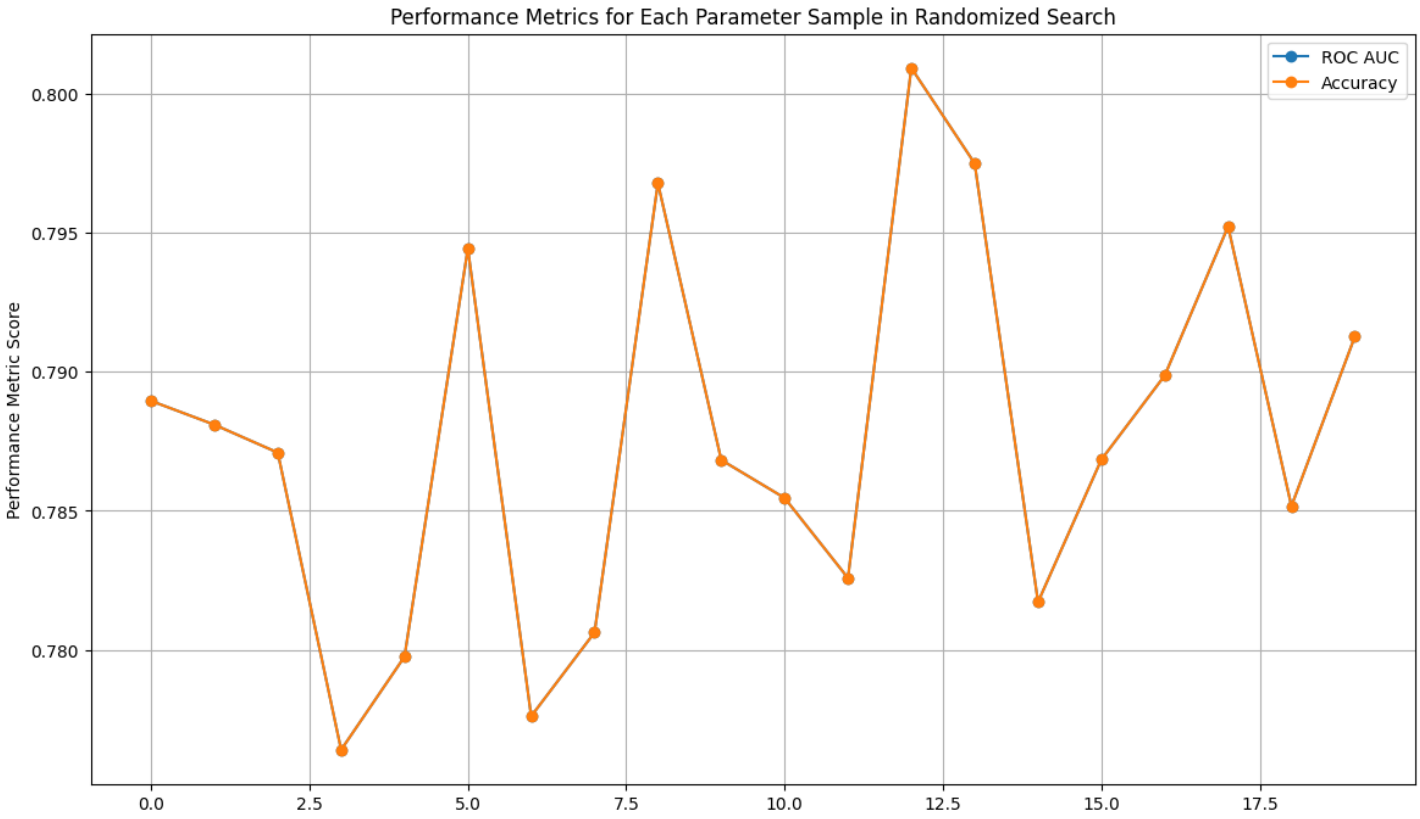
In [26]:

```
param_grid = {
    'learning_rate': [0.1],
    'max_depth': [3],
    'n_estimators': [100]
}

xgb_model = XGBClassifier(
    eval_metric='logloss',
    scale_pos_weight=len(y_train[y_train == 0]) / len(y_train[y_train == 1])
)

grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    cv=3,  # using 3-fold for quicker testing
    scoring='roc_auc',
    n_jobs=-1
)

grid_search.fit(X_train, y_train)
print("Best parameters found:", grid_search.best_params_)
```

```
Best parameters found: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}
```

In [27]:

```
results = pd.DataFrame(random_search.cv_results_)

# Plot performance metrics for each parameter sample
plt.figure(figsize=(14, 8))

# Plot each metric across the parameter samples (mean scores)
plt.plot(results.index, results['mean_test_score'], label='ROC AUC', marker='o', linesty
le='-')
plt.plot(results.index, results['mean_test_score'], label='Accuracy', marker='o', linest
yle='-')

plt.xlabel("Parameter Sample Index")
plt.ylabel("Performance Metric Score")
plt.title("Performance Metrics for Each Parameter Sample in Randomized Search")
plt.legend(loc="best")
plt.grid(True)
plt.show()
```



Performance on Test Data

In [28]:

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [29]:

```
xgb_model = XGBClassifier(eval_metric='logloss')
xgb_model.fit(X_train, y_train)
```

Out[29]:

▼ XGBClassifier i

```
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='logloss',
              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=No
```

In [30]:

```
# Loading loan_testdata
test_data = pd.read_excel('/content/Loan_TestData.xlsx')
for column in test_data.columns:
    if column in ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Cred
it_History']:
        test_data[column] = test_data[column].fillna(test_data[column].mode()[0])
    elif test_data[column].dtype in ['float64', 'int64']:
        test_data[column] = test_data[column].fillna(test_data[column].mean())

# Checking if there are any remaining missing values
missing_values_after_filling = test_data.isnull().sum()
print("Missing values after filling:", missing_values_after_filling)

# Applying the same preprocessing as on the training data-Feature engineering, encoding,
and scaling



test_data['Total_Income'] = test_data['Applicant_Income'] + test_data['Coapplicant_Income
']
test_data['Loan_Income_Ratio'] = test_data['Loan_Amount'] / test_data['Total_Income']
test_data['Income_Per_Dependent'] = test_data['Applicant_Income'] / (test_data['Dependent
s'] + 1)

for column, le in label_encoders.items():
    if column in test_data.columns:
        test_data[column] = le.transform(test_data[column])


# Scale numerical features (using the scaler fitted on the training data)
test_data[numerical_columns] = scaler.transform(test_data[numerical_columns])

for column in test_data.select_dtypes(include=['object']).columns:
    test_data[column] = test_data[column].astype('category')


#test_data.to_csv("test_processed_data.csv", index=False)
```

```
Missing values after filling: Gender                0
```

```
Married               0
Dependents            0
Education             0
Self_Employed         0
Applicant_Income      0
Coapplicant_Income    0
Loan_Amount           0
Term                  0
Credit_History        0
Area                  0
dtype: int64
```

In [31]:

```python
# Predict on test data
test_predictions = xgb_model.predict(test_data)
test_probabilities = xgb_model.predict_proba(test_data)[:, 1]
```

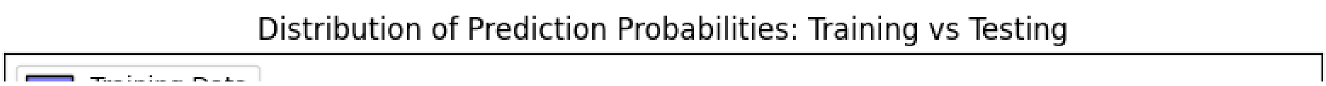
In [32]:

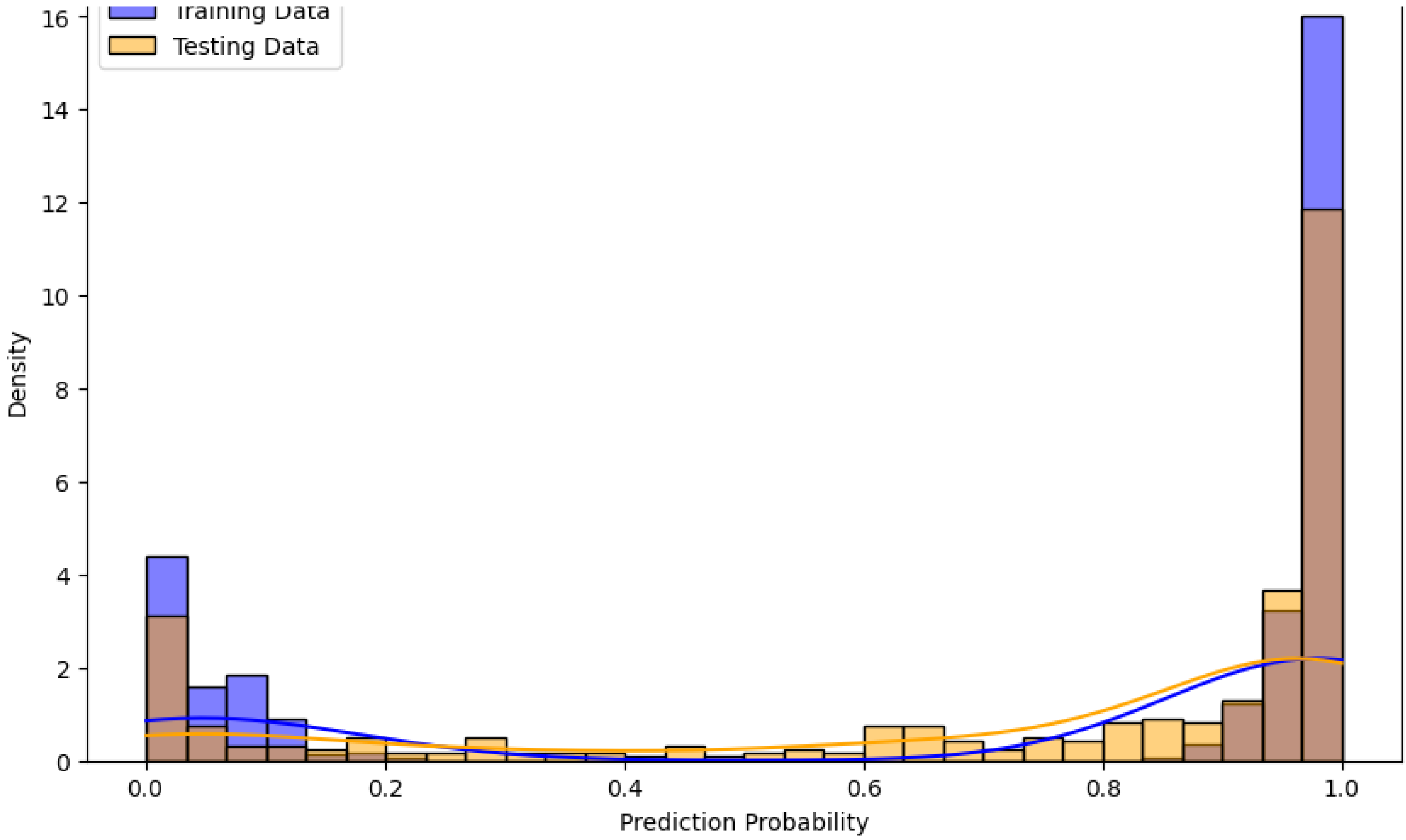
```python
results_df = pd.DataFrame({
    'Loan_ID': test_data['Loan_ID'] if 'Loan_ID' in test_data.columns else range(1, len(
test_predictions) + 1),
    'Predicted_Status': test_predictions,
    'Prediction_Probability': test_probabilities
})

print("Predictions saved to 'test_predictions.csv'")
print(results_df.head(20))
```

```
Predictions saved to 'test_predictions.csv'
    Loan_ID  Predicted_Status  Prediction_Probability
0         1                 1                0.991393
1         2                 1                0.962995
2         3                 1                0.985601
3         4                 1                0.974410
4         5                 0                0.175209
5         6                 1                0.984210
6         7                 1                0.973798
7         8                 0                0.003397
8         9                 1                0.977138
9        10                 1                0.999009
10       11                 1                0.971889
11       12                 0                0.036220
12       13                 1                0.865060
13       14                 0                0.027539
14       15                 1                0.716061
15       16                 1                0.996717
16       17                 1                0.874282
17       18                 1                0.993015
18       19                 1                0.746684
19       20                 1                0.935225
```

In [33]:

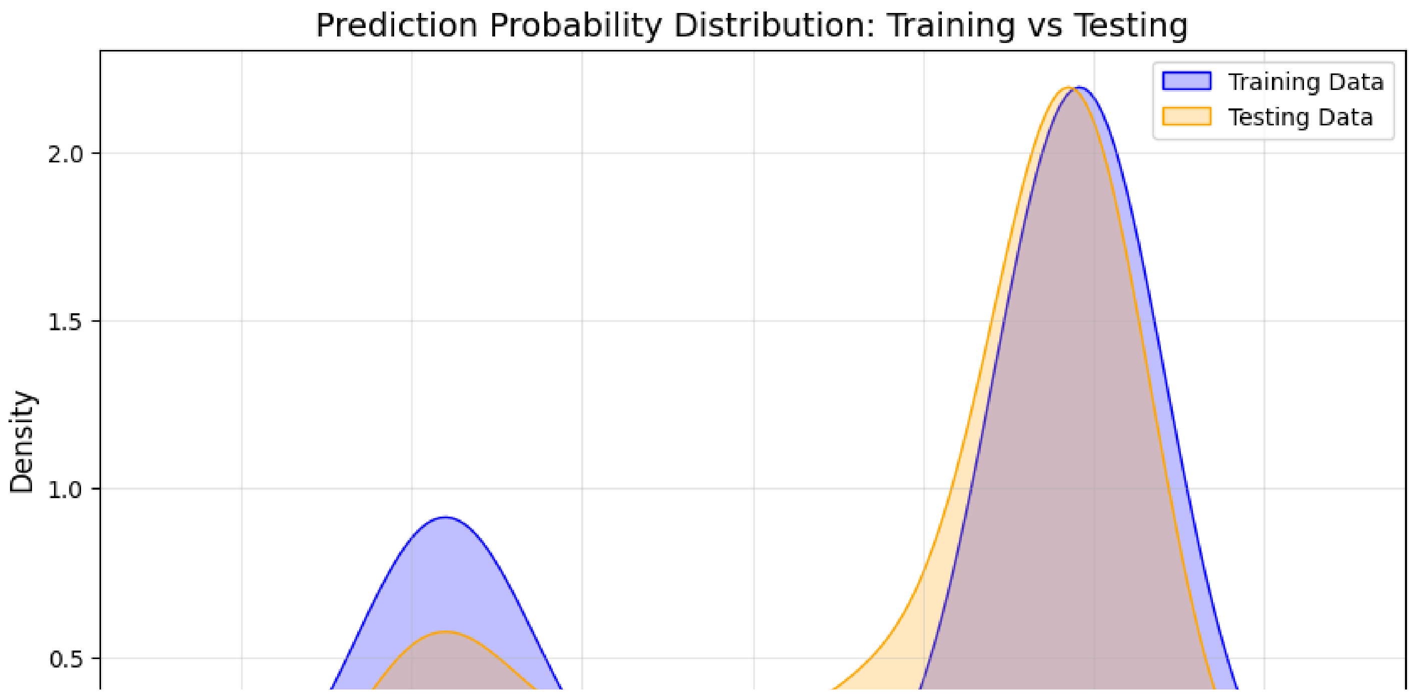
```python
# Assuming training probabilities are available from the model
train_probabilities = xgb_model.predict_proba(X_train)[:, 1]

plt.figure(figsize=(10, 6))
sns.histplot(train_probabilities, bins=30, kde=True, color='blue', label='Training Data',
stat='density')
sns.histplot(test_probabilities, bins=30, kde=True, color='orange', label='Testing Data',
stat='density')
plt.title("Distribution of Prediction Probabilities: Training vs Testing")
plt.xlabel("Prediction Probability")
plt.ylabel("Density")
plt.legend()
plt.show()
```

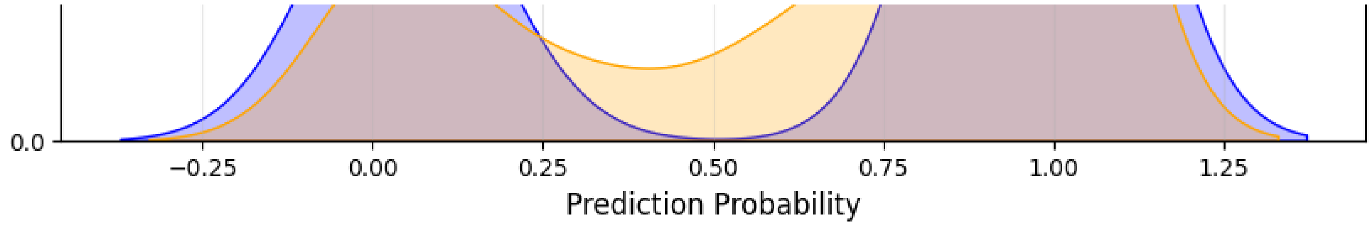




In [34]:

```
data = pd.DataFrame({
    'Probability': list(train_probabilities) + list(test_probabilities),
    'Dataset': ['Training'] * len(train_probabilities) + ['Testing'] * len(test_probabil
ities)
})

# Plotting side-by-side density plots
plt.figure(figsize=(10, 6))
sns.kdeplot(data=data[data['Dataset'] == 'Training']['Probability'], color='blue', fill=
True, label='Training Data')
sns.kdeplot(data=data[data['Dataset'] == 'Testing']['Probability'], color='orange', fill=
True, label='Testing Data')
plt.title("Prediction Probability Distribution: Training vs Testing", fontsize=14)
plt.xlabel("Prediction Probability", fontsize=12)
plt.ylabel("Density", fontsize=12)
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```





Implementing GNN

In [35]:

```
X = torch.tensor(train_data.drop(columns=['Status']).values, dtype=torch.float)
y = torch.tensor(train_data['Status'].values, dtype=torch.long)

# Defining edges based on feature similarity
edge_index = []
for i in range(len(train_data)):
    for j in range(i+1, len(train_data)):
        if train_data['Credit_History'].iloc[i] == train_data['Credit_History'].iloc[j]:
            edge_index.append([i, j])
            edge_index.append([j, i])

edge_index = torch.tensor(edge_index, dtype=torch.long).t().contiguous()
data = Data(x=X, edge_index=edge_index, y=y)
num_nodes = len(train_data)
train_size = int(0.8 * num_nodes)  # 80% for training
train_mask = torch.zeros(num_nodes, dtype=torch.bool)
test_mask = torch.zeros(num_nodes, dtype=torch.bool)

# Randomly assigning 80% of nodes to training and the rest to testing
train_indices = np.random.choice(num_nodes, train_size, replace=False)
train_mask[train_indices] = True
test_mask[~train_mask] = True
data.train_mask = train_mask
data.test_mask = test_mask
class GNN(torch.nn.Module):
    def __init__(self):
        super(GNN, self).__init__()
        self.conv1 = GCNConv(X.shape[1], 16)  # Input features to hidden layer
        self.conv2 = GCNConv(16, 2)           # Hidden layer to output layer (2 classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)

model = GNN()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)
loss_fn = torch.nn.CrossEntropyLoss()

model.train()
for epoch in range(100):
    optimizer.zero_grad()
    out = model(data)
    loss = loss_fn(out[data.train_mask], data.y[data.train_mask])
    loss.backward()
    optimizer.step()
    print(f"Epoch {epoch+1}, Loss: {loss.item():.4f}")

model.eval()
_, pred = model(data).max(dim=1)
correct = int((pred[data.test_mask] == data.y[data.test_mask]).sum())
accuracy = correct / int(data.test_mask.sum())
print(f'Accuracy: {accuracy:.4f}')
```

```
Epoch 1, Loss: 5.9384
Epoch 2, Loss: 2.6162
Epoch 3, Loss: 1.4641
```

```
Epoch 4, Loss: 1.5420
Epoch 5, Loss: 0.7528
Epoch 6, Loss: 0.6395
Epoch 7, Loss: 0.7053
Epoch 8, Loss: 0.6388
Epoch 9, Loss: 0.6176
Epoch 10, Loss: 0.6462
Epoch 11, Loss: 0.6091
Epoch 12, Loss: 0.6456
Epoch 13, Loss: 0.6188
Epoch 14, Loss: 0.6222
Epoch 15, Loss: 0.6251
Epoch 16, Loss: 0.6040
Epoch 17, Loss: 0.6213
Epoch 18, Loss: 0.6012
Epoch 19, Loss: 0.6354
Epoch 20, Loss: 0.6196
Epoch 21, Loss: 0.6109
Epoch 22, Loss: 0.6309
Epoch 23, Loss: 0.5962
Epoch 24, Loss: 0.6144
Epoch 25, Loss: 0.6003
Epoch 26, Loss: 0.5970
Epoch 27, Loss: 0.6070
Epoch 28, Loss: 0.5894
Epoch 29, Loss: 0.6038
Epoch 30, Loss: 0.5979
Epoch 31, Loss: 0.5883
Epoch 32, Loss: 0.6029
Epoch 33, Loss: 0.5906
Epoch 34, Loss: 0.5862
Epoch 35, Loss: 0.5972
Epoch 36, Loss: 0.5857
Epoch 37, Loss: 0.5819
Epoch 38, Loss: 0.5913
Epoch 39, Loss: 0.5830
Epoch 40, Loss: 0.5770
Epoch 41, Loss: 0.5859
Epoch 42, Loss: 0.5822
Epoch 43, Loss: 0.5725
Epoch 44, Loss: 0.5805
Epoch 45, Loss: 0.5823
Epoch 46, Loss: 0.5697
Epoch 47, Loss: 0.5739
Epoch 48, Loss: 0.5806
Epoch 49, Loss: 0.5672
Epoch 50, Loss: 0.5649
Epoch 51, Loss: 0.5664
Epoch 52, Loss: 0.6057
Epoch 53, Loss: 0.7186
Epoch 54, Loss: 0.5933
Epoch 55, Loss: 0.5633
Epoch 56, Loss: 0.6209
Epoch 57, Loss: 0.5988
Epoch 58, Loss: 0.5685
Epoch 59, Loss: 0.5564
Epoch 60, Loss: 0.5788
Epoch 61, Loss: 0.6035
Epoch 62, Loss: 0.5631
Epoch 63, Loss: 0.5530
Epoch 64, Loss: 0.5737
Epoch 65, Loss: 0.5803
Epoch 66, Loss: 0.5847
Epoch 67, Loss: 0.5547
Epoch 68, Loss: 0.5482
Epoch 69, Loss: 0.5641
Epoch 70, Loss: 0.5663
Epoch 71, Loss: 0.5609
Epoch 72, Loss: 0.5452
Epoch 73, Loss: 0.5442
Epoch 74, Loss: 0.5545
Epoch 75, Loss: 0.5563
```

```
Epoch 76, Loss: 0.5540
Epoch 77, Loss: 0.5425
Epoch 78, Loss: 0.5381
Epoch 79, Loss: 0.5411
Epoch 80, Loss: 0.5458
Epoch 81, Loss: 0.5505
Epoch 82, Loss: 0.5446
Epoch 83, Loss: 0.5395
Epoch 84, Loss: 0.5345
Epoch 85, Loss: 0.5327
Epoch 86, Loss: 0.5335
Epoch 87, Loss: 0.5356
Epoch 88, Loss: 0.5387
Epoch 89, Loss: 0.5388
Epoch 90, Loss: 0.5404
Epoch 91, Loss: 0.5376
Epoch 92, Loss: 0.5371
Epoch 93, Loss: 0.5340
Epoch 94, Loss: 0.5331
Epoch 95, Loss: 0.5310
Epoch 96, Loss: 0.5308
Epoch 97, Loss: 0.5299
Epoch 98, Loss: 0.5313
Epoch 99, Loss: 0.5320
Epoch 100, Loss: 0.5375
Accuracy: 0.8130
```

In [36]:

```
# Ensuring all data is numeric
for column in test_data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    test_data[column] = le.fit_transform(test_data[column])

# Ensuring numeric conversion and filling missing values
test_data = test_data.apply(pd.to_numeric, errors='coerce')
test_data = test_data.fillna(0)

test_X = torch.tensor(test_data.values, dtype=torch.float)
test_edge_index = []

# Connect nodes with the same Credit_History (similar to training)
for i in range(len(test_data)):
    for j in range(i + 1, len(test_data)):
        if test_data.iloc[i]['Credit_History'] == test_data.iloc[j]['Credit_History']:
            test_edge_index.append([i, j])
            test_edge_index.append([j, i])

test_edge_index = torch.tensor(test_edge_index, dtype=torch.long).t().contiguous()

test_data_object = Data(x=test_X, edge_index=test_edge_index)

# Apply the trained GNN model
model.eval()  # Set the model to evaluation mode
with torch.no_grad():
    test_out = model(test_data_object)
    test_pred = test_out.max(dim=1)[1]
    test_probabilities = torch.softmax(test_out, dim=1)[:, 1]


test_data['Predicted_Status'] = test_pred.numpy()
test_data['Prediction_Probability'] = test_probabilities.numpy()

results_df = pd.DataFrame({
    'Loan_ID': range(1, len(test_pred) + 1),
    'Predicted_Status': test_pred.numpy(),
    'Prediction_Probability': test_probabilities.numpy()
})


results_df.to_csv('test_gnn_predictions.csv', index=False)
```

```
print("Predictions saved to 'test_gnn_predictions.csv'")
print(results_df.head(20))
```

```
Predictions saved to 'test_gnn_predictions.csv'
    Loan_ID  Predicted_Status  Prediction_Probability
0         1                 1                0.590154
1         2                 1                0.590154
2         3                 1                0.590154
3         4                 1                0.590154
4         5                 1                0.590154
5         6                 1                0.590154
6         7                 1                0.590154
7         8                 0                0.291639
8         9                 1                0.590154
9        10                 1                0.590154
10       11                 1                0.590154
11       12                 1                0.590154
12       13                 1                0.590154
13       14                 0                0.291639
14       15                 1                0.590154
15       16                 1                0.590154
16       17                 1                0.590154
17       18                 1                0.590154
18       19                 1                0.590154
19       20                 1                0.590154
```

Stacked Ensemble of Tree-Based Models with Meta-Learning

In [37]:

```
X = train_data.drop(columns=['Status'])
y = train_data['Status']


X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

# Preparing out-of-fold predictions for meta-learner
n_splits = 5
kf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)

# Placeholders for out-of-fold predictions
train_meta = np.zeros((X_train.shape[0], 3))  # 3 models
val_meta = np.zeros((X_val.shape[0], 3))

# Defining base learners
xgb_model = xgb.XGBClassifier(eval_metric='logloss', random_state=42)
lgb_model = lgb.LGBMClassifier(random_state=42)
cat_model = CatBoostClassifier(verbose=0, random_state=42)


for fold, (train_idx, val_idx) in enumerate(kf.split(X_train, y_train)):
    print(f"Training fold {fold + 1}/{n_splits}")

    X_tr, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_tr, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    # XGBoost
    xgb_model.fit(X_tr, y_tr)
    train_meta[val_idx, 0] = xgb_model.predict_proba(X_val_fold)[:, 1]
    val_meta[:, 0] += xgb_model.predict_proba(X_val)[:, 1] / n_splits

    # LightGBM
    lgb_model.fit(X_tr, y_tr)
    train_meta[val_idx, 1] = lgb_model.predict_proba(X_val_fold)[:, 1]
    val_meta[:, 1] += lgb_model.predict_proba(X_val)[:, 1] / n_splits

    # CatBoost
    cat_model.fit(X_tr, y_tr)
    train_meta[val_idx, 2] = cat_model.predict_proba(X_val_fold)[:, 1]
```

```python
    val_meta[:, 2] += cat_model.predict_proba(X_val)[:, 1] / n_splits

meta_learner = LogisticRegression(random_state=42)
meta_learner.fit(train_meta, y_train)


val_pred = meta_learner.predict(val_meta)
val_proba = meta_learner.predict_proba(val_meta)[:, 1]
```

```
Training fold 1/5
[LightGBM] [Info] Number of positive: 269, number of negative: 123
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.0
00677 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 704
[LightGBM] [Info] Number of data points in the train set: 392, number of used features: 1
4
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.686224 -> initscore=0.782527
[LightGBM] [Info] Start training from score 0.782527
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Training fold 2/5
[LightGBM] [Info] Number of positive: 269, number of negative: 124
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.0
00104 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 706
[LightGBM] [Info] Number of data points in the train set: 393, number of used features: 1
4
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.684478 -> initscore=0.774430
[LightGBM] [Info] Start training from score 0.774430
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Training fold 3/5
[LightGBM] [Info] Number of positive: 270, number of negative: 123
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.0
00103 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 707
[LightGBM] [Info] Number of data points in the train set: 393, number of used features: 1
4
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.687023 -> initscore=0.786238
[LightGBM] [Info] Start training from score 0.786238
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Training fold 4/5
[LightGBM] [Info] Number of positive: 270, number of negative: 123
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.0
00113 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 704
[LightGBM] [Info] Number of data points in the train set: 393, number of used features: 1
4
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.687023 -> initscore=0.786238
[LightGBM] [Info] Start training from score 0.786238
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Training fold 5/5
[LightGBM] [Info] Number of positive: 270, number of negative: 123
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.0
00120 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 709
[LightGBM] [Info] Number of data points in the train set: 393, number of used features: 1
4
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.687023 -> initscore=0.786238
[LightGBM] [Info] Start training from score 0.786238
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

In [38]:

```python
# Evaluating the stacked model
accuracy = accuracy_score(y_val, val_pred)
roc_auc = roc_auc_score(y_val, val_proba)
print(f"Stacked Model Accuracy: {accuracy:.4f}")
print(f"Stacked Model ROC AUC: {roc_auc:.4f}")
```

```
Stacked Model Accuracy: 0.8455
Stacked Model ROC AUC: 0.8517
```

In [39]:

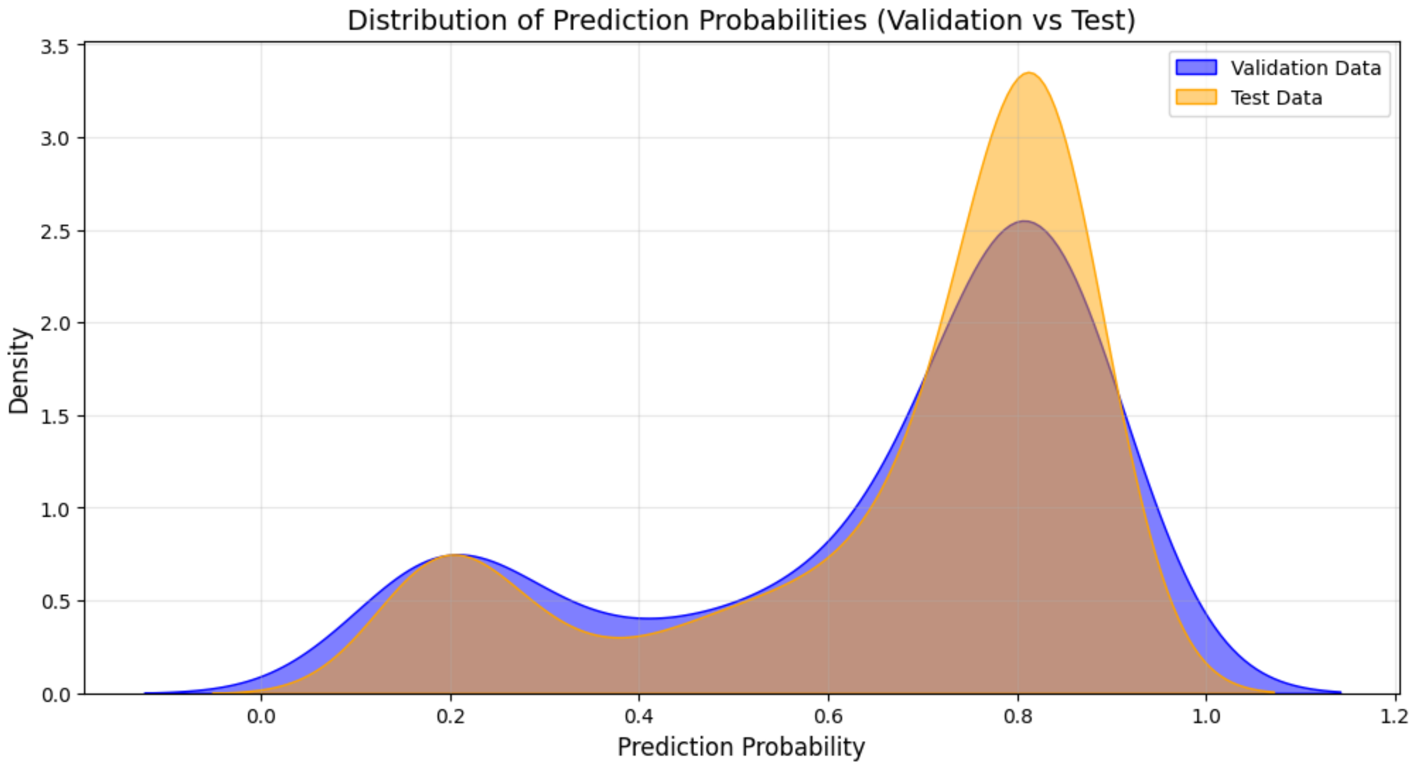
```
test_data_features = test_data[X.columns]

# Generating meta-features for the test data
test_meta = np.zeros((test_data_features.shape[0], 3))

test_meta[:, 0] = xgb_model.predict_proba(test_data_features)[:, 1]  # XGBoost probabilit
ies
test_meta[:, 1] = lgb_model.predict_proba(test_data_features)[:, 1]  # LightGBM probabili
ties
test_meta[:, 2] = cat_model.predict_proba(test_data_features)[:, 1]  # CatBoost probabili
ties


test_pred = meta_learner.predict(test_meta)
test_proba = meta_learner.predict_proba(test_meta)[:, 1]


results_df = pd.DataFrame({
    'Loan_ID': range(1, len(test_pred) + 1),
    'Predicted_Status': test_pred,
    'Prediction_Probability': test_proba
})

results_df.to_csv('stacked_test_predictions.csv', index=False)

print("Predictions saved to 'stacked_test_predictions.csv'")
print(results_df.head(20))
```

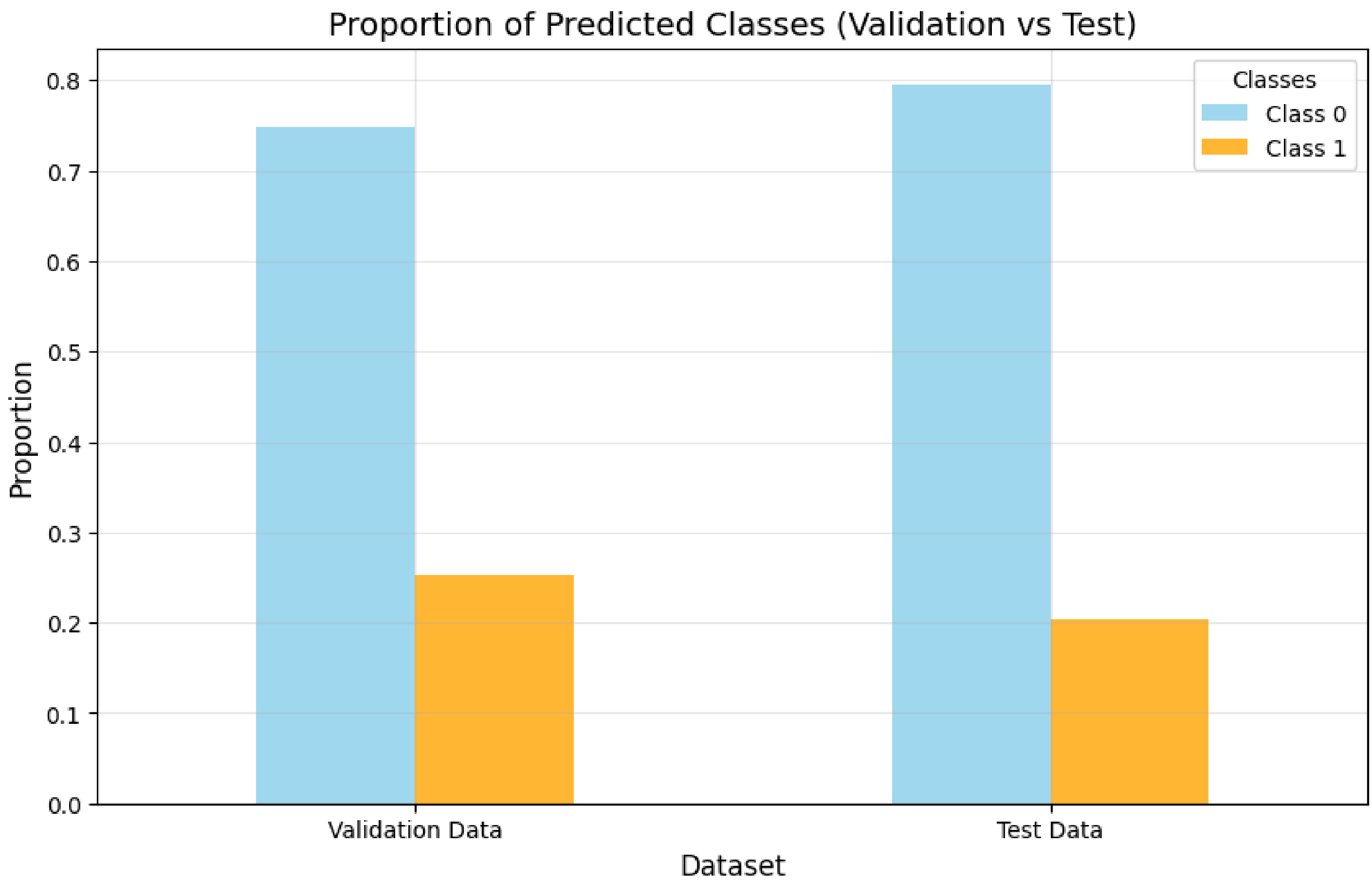
```
Predictions saved to 'stacked_test_predictions.csv'
    Loan_ID  Predicted_Status  Prediction_Probability
0         1                 1                0.824445
1         2                 1                0.837167
2         3                 1                0.798786
3         4                 1                0.805981
4         5                 0                0.459993
5         6                 1                0.834469
6         7                 1                0.724697
7         8                 0                0.185468
8         9                 1                0.865360
9        10                 1                0.862528
10       11                 1                0.700933
11       12                 1                0.775862
12       13                 1                0.556740
13       14                 0                0.356299
14       15                 1                0.775447
15       16                 1                0.837474
16       17                 1                0.535851
17       18                 1                0.813614
18       19                 1                0.767687
19       20                 1                0.659965
```

In [40]:

```
#Distribution of Prediction Probabilities
plt.figure(figsize=(12, 6))
sns.kdeplot(val_proba, label='Validation Data', fill=True, color='blue', alpha=0.5)
sns.kdeplot(test_proba, label='Test Data', fill=True, color='orange', alpha=0.5)
plt.title('Distribution of Prediction Probabilities (Validation vs Test)', fontsize=14)
plt.xlabel('Prediction Probability', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.legend()
plt.grid(alpha=0.3)
plt.show()

#Proportion of Predicted Classes
val_pred_counts = pd.Series(val_pred).value_counts(normalize=True)
test_pred_counts = pd.Series(test_pred).value_counts(normalize=True)


pred_df = pd.DataFrame({
```

```
    'Validation Data': val_pred_counts,
    'Test Data': test_pred_counts
}).T
pred_df.columns = ['Class 0', 'Class 1']

pred_df.plot(kind='bar', figsize=(10, 6), color=['skyblue', 'orange'], alpha=0.8)
plt.title('Proportion of Predicted Classes (Validation vs Test)', fontsize=14)
plt.ylabel('Proportion', fontsize=12)
plt.xlabel('Dataset', fontsize=12)
plt.xticks(rotation=0)
plt.grid(alpha=0.3)
plt.legend(title='Classes')
plt.show()
```





Feature Augmentation with K-Means and XGBoost

Feature Augmentation with K-Means and XGBoost

In [41]:

```
X = train_data.drop(columns=['Status'])
y = train_data['Status']

#Applying K-Means Clustering
n_clusters = 5
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
cluster_labels = kmeans.fit_predict(X)

X['Cluster'] = cluster_labels

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

# Train XGBoost Model on Augmented Data
xgb_model = XGBClassifier(eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)

y_pred = xgb_model.predict(X_val)
y_pred_proba = xgb_model.predict_proba(X_val)[:, 1]

accuracy = accuracy_score(y_val, y_pred)
roc_auc = roc_auc_score(y_val, y_pred_proba)

print(f"XGBoost with Clustering - Accuracy: {accuracy:.4f}")
print(f"XGBoost with Clustering - ROC AUC: {roc_auc:.4f}")
```

```
XGBoost with Clustering - Accuracy: 0.8455
XGBoost with Clustering - ROC AUC: 0.8588
```

In [42]:

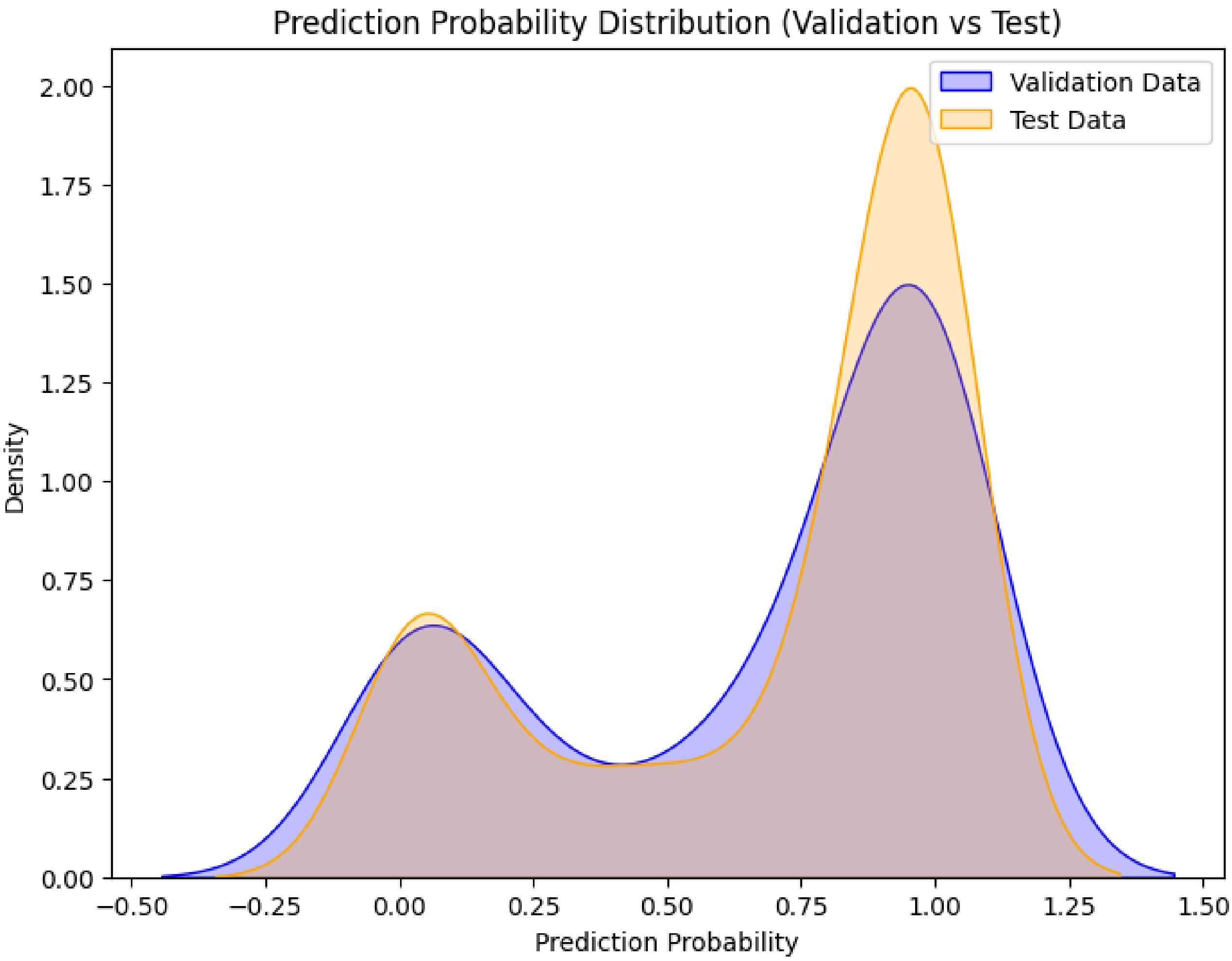
```
test_features = test_data[X.columns.drop('Cluster', errors='ignore')]

test_features = test_features.copy()
test_features.loc[:, 'Cluster'] = kmeans.predict(test_features)

y_test_pred = xgb_model.predict(test_features)
y_test_proba = xgb_model.predict_proba(test_features)[:, 1]


results_df = pd.DataFrame({
    'Loan_ID': range(1, len(y_test_pred) + 1),
    'Predicted_Status': y_test_pred,
    'Prediction_Probability': y_test_proba
})

results_df.to_csv('test_xgb_with_clustering_predictions.csv', index=False)

print("Prediction probabilities are")
print(results_df.head(20))
```

```
Prediction probabilities are
    Loan_ID  Predicted_Status  Prediction_Probability
0         1                 1                0.986799
1         2                 1                0.995132
2         3                 1                0.912581
3         4                 1                0.941941
4         5                 0                0.248625
5         6                 1                0.959468
6         7                 1                0.783155
7         8                 0                0.002060
8         9                 1                0.999143
9        10                 1                0.998671
10       11                 1                0.832006
11       12                 0                0.249193
12       13                 1                0.711585
13       14                 0                0.256563
14       15                 1                0.930449
15       16                 1                0.999119
```

```
16        17                   0                0.051074
17        18                   1                0.990446
18        19                   1                0.894199
19        20                   0                0.370467
```

In [43]:

```
val_data_df = pd.DataFrame({
    'Predicted_Status': y_pred,
    'Prediction_Probability': y_pred_proba,
    'Dataset': 'Validation'
})

test_data_df = pd.DataFrame({
    'Predicted_Status': y_test_pred,
    'Prediction_Probability': y_test_proba,
    'Dataset': 'Test'
})

plt.figure(figsize=(8, 6))
sns.kdeplot(data=val_data_df['Prediction_Probability'], label='Validation Data', fill=True, color='blue')
sns.kdeplot(data=test_data_df['Prediction_Probability'], label='Test Data', fill=True, color='orange')
plt.title('Prediction Probability Distribution (Validation vs Test)')
plt.xlabel('Prediction Probability')
plt.ylabel('Density')
plt.legend()
plt.show()
```



Steps to set up the web page

In [44]:

```
exclude_columns = ['Total_Income', 'Loan_Income_Ratio', 'Income_Per_Dependent']

X_train = train_data.drop(columns=['Status'] + exclude_columns)
```

```
y_train = train_data['Status']

xgb_model = XGBClassifier(eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)

joblib.dump(xgb_model, "xgboost_model.pkl")
```

Out[44]:

```
['xgboost_model.pkl']
```

In [45]:

```
%%writefile app.py

# Loaded the saved XGBoost model
xgb_model = joblib.load("xgboost_model.pkl")

st.title("Loan Approval Prediction")

gender = st.selectbox("Gender", ["Male", "Female"])
married = st.selectbox("Married", ["Yes", "No"])
dependents = st.selectbox("Dependents", ["0", "1", "2", "3+"])
education = st.selectbox("Education", ["Graduate", "Not Graduate"])
self_employed = st.selectbox("Self Employed", ["Yes", "No"])
applicant_income = st.number_input("Applicant Income", min_value=0)
coapplicant_income = st.number_input("Coapplicant Income", min_value=0)
loan_amount = st.number_input("Loan Amount", min_value=0)
loan_amount_term = st.number_input("Loan Amount Term", min_value=0)
credit_history = st.selectbox("Credit History", [0, 1])
property_area = st.selectbox("Property Area", ["Urban", "Semiurban", "Rural"])


gender_map = {"Male": 0, "Female": 1}
married_map = {"No": 0, "Yes": 1}
dependents_map = {"0": 0, "1": 1, "2": 2, "3+": 3}
education_map = {"Not Graduate": 0, "Graduate": 1}
self_employed_map = {"No": 0, "Yes": 1}
property_area_map = {"Rural": 0, "Semiurban": 1, "Urban": 2}


input_data = np.array([
    gender_map[gender],
    married_map[married],
    dependents_map[dependents],
    education_map[education],
    self_employed_map[self_employed],
    applicant_income,
    coapplicant_income,
    loan_amount,
    loan_amount_term,
    credit_history,
    property_area_map[property_area]
]).reshape(1, -1)

# Predicting loan approval status when the button is pressed
if st.button("Predict Loan Approval"):
    prediction = xgb_model.predict(input_data)[0]
    prediction_proba = xgb_model.predict_proba(input_data)[0, 1]


    if prediction == 1:
        st.success(f"Loan Approved! Approval Probability: {prediction_proba:.2f}")
    else:
        st.error(f"Loan Not Approved. Approval Probability: {prediction_proba:.2f}")
```

```
Writing app.py
```

In [46]:

```
!nohup streamlit run app.py &
```

```
nohup: appending output to 'nohup.out'
```

```
nohup: appending output to 'nohup.out'
```

In [47]:

```python
# Authenticating ngrok with my authtoken
ngrok.set_auth_token("2o0iDYvFWr9dAPTUCath3ib8BT3_3FwkeXAhRcL9fwjNr1k9i")
```

In [48]:

```python
# Starting ngrok to tunnel the Streamlit app
public_url = ngrok.connect(8501)
print(f"Streamlit app is live at: {public_url}")
```

```
Streamlit app is live at: NgrokTunnel: "https://c258-35-230-83-65.ngrok-free.app" -> "htt
p://localhost:8501"
```