

```
In [5]: pip install pandas
```

```
Requirement already satisfied: pandas in c:  
\users\hp\appdata\local\programs\python\python313\lib\site-packages (2.3.3)  
Requirement already satisfied: numpy>=1.26.0 in c:  
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.  
Requirement already satisfied: python-dateutil>=2.8.2 in c:  
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas)  
(2.9.0.post0)  
Requirement already satisfied: pytz>=2020.1 in c:  
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas) (26  
Requirement already satisfied: tzdata>=2022.7 in c:  
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas) (26  
Requirement already satisfied: six>=1.5 in c:  
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from python-  
dateutil>=2.8.2->pandas) (1.17.0)  
Note: you may need to restart the kernel to use updated packages.  
[notice] A new release of pip is available: 25.1.1 -> 25.3  
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
In [7]: import pandas as pd
```

```
In [8]: pip install seaborn
```

```
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from seaborn) (:)
Requirement already satisfied: pandas>=1.2 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from seaborn) (:)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from seaborn) (:)
Requirement already satisfied: contourpy>=1.0.1 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.3)
Requirement already satisfied: cycler>=0.10 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.9)
Requirement already satisfied: packaging>=20.0 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (25.0)
Requirement already satisfied: pillow>=8 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (12.0.0)
Requirement already satisfied: pyparsing>=3 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.3.1)
Requirement already satisfied: python-dateutil>=2.7 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas>=1.1->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas>=1.1->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
In [9]: import seaborn as sns
```

```
In [10]: pip install numpy
```

```
Requirement already satisfied: numpy in c:
\users\hp\appdata\local\programs\python\python313\lib\site-packages (2.3.4)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
In [11]: import numpy as np
```

In [12]: `pip install scikit-learn`

```
Collecting scikit-learn
  Downloading scikit_learn-1.8.0-cp313-cp313-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: numpy>=1.24.1 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn<2.3.4)
Collecting scipy>=1.10.0 (from scikit-learn)
  Downloading scipy-1.16.3-cp313-cp313-win_amd64.whl.metadata (60 kB)
Collecting joblib>=1.3.0 (from scikit-learn)
  Downloading joblib-1.5.3-py3-none-any.whl.metadata (5.5 kB)
Collecting threadpoolctl>=3.2.0 (from scikit-learn)
  Downloading threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.8.0-cp313-cp313-win_amd64.whl (8.0 MB)
----- 0.0/8.0 MB ? eta -:-:--
----- 0.3/8.0 MB ? eta -:-:--
----- 1.3/8.0 MB 3.3 MB/s eta 0:00:03
----- 1.6/8.0 MB 3.2 MB/s eta 0:00:03
----- 2.6/8.0 MB 3.2 MB/s eta 0:00:02
----- 3.9/8.0 MB 3.9 MB/s eta 0:00:02
----- 5.0/8.0 MB 4.1 MB/s eta 0:00:01
----- 5.8/8.0 MB 4.0 MB/s eta 0:00:01
----- 7.1/8.0 MB 4.4 MB/s eta 0:00:01
----- 8.0/8.0 MB 4.6 MB/s eta 0:00:00
Downloading joblib-1.5.3-py3-none-any.whl (309 kB)
Downloading scipy-1.16.3-cp313-cp313-win_amd64.whl (38.5 MB)
----- 0.0/38.5 MB ? eta -:-:--
----- 1.6/38.5 MB 8.0 MB/s eta 0:00:05
----- 3.4/38.5 MB 8.4 MB/s eta 0:00:05
----- 5.0/38.5 MB 8.1 MB/s eta 0:00:05
----- 7.1/38.5 MB 8.5 MB/s eta 0:00:04
----- 9.2/38.5 MB 8.8 MB/s eta 0:00:04
----- 11.0/38.5 MB 8.7 MB/s eta 0:00:04
----- 12.8/38.5 MB 8.7 MB/s eta 0:00:03
----- 14.7/38.5 MB 8.7 MB/s eta 0:00:03
----- 16.5/38.5 MB 8.7 MB/s eta 0:00:03
----- 18.1/38.5 MB 8.7 MB/s eta 0:00:03
----- 18.9/38.5 MB 8.5 MB/s eta 0:00:03
----- 19.1/38.5 MB 7.7 MB/s eta 0:00:03
----- 21.0/38.5 MB 7.7 MB/s eta 0:00:03
----- 22.8/38.5 MB 7.8 MB/s eta 0:00:03
----- 24.1/38.5 MB 7.6 MB/s eta 0:00:02
----- 24.6/38.5 MB 7.4 MB/s eta 0:00:02
----- 25.7/38.5 MB 7.2 MB/s eta 0:00:02
----- 26.7/38.5 MB 7.1 MB/s eta 0:00:02
----- 28.0/38.5 MB 7.0 MB/s eta 0:00:02
----- 29.6/38.5 MB 7.1 MB/s eta 0:00:02
----- 31.5/38.5 MB 7.2 MB/s eta 0:00:01
----- 32.8/38.5 MB 7.1 MB/s eta 0:00:01
----- 34.3/38.5 MB 7.1 MB/s eta 0:00:01
----- 35.7/38.5 MB 7.2 MB/s eta 0:00:01
----- 37.2/38.5 MB 7.1 MB/s eta 0:00:01
----- 38.3/38.5 MB 7.1 MB/s eta 0:00:01
----- 38.5/38.5 MB 7.0 MB/s eta 0:00:00
Downloading threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
```

```
----- 1/4 [scipy]
```



```
----- 3/4 [scikit-learn]
```

Successfully installed joblib-1.5.3 scikit-learn-1.8.0 scipy-1.16.3 threadpoolctl-3.0
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip

In [13]: `import sklearn`

In [2]: `pip install matplotlib`

```
Collecting matplotlib
  Downloading matplotlib-3.10.8-cp313-cp313-win_amd64.whl.metadata (52 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.3-cp313-cp313-win_amd64.whl.metadata (5.5 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.61.1-cp313-cp313-win_amd64.whl.metadata (116 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.9-cp313-cp313-win_amd64.whl.metadata (6.4 kB)
Requirement already satisfied: numpy>=1.23 in c:
  \users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib)
(2.3.4)
Requirement already satisfied: packaging>=20.0 in c:
  \users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib)
(25.0)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-12.0.0-cp313-cp313-win_amd64.whl.metadata (9.0 kB)
Collecting pyparsing>=3 (from matplotlib)
  Downloading pyparsing-3.3.1-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: python-dateutil>=2.7 in c:
  \users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:
  \users\hp\appdata\local\programs\python\python313\lib\site-packages (from python-
dateutil>=2.7->matplotlib) (1.17.0)
  Downloading matplotlib-3.10.8-cp313-cp313-win_amd64.whl (8.1 MB)
----- 0.0/8.1 MB ? eta -:-:--
----- 0.8/8.1 MB 5.3 MB/s eta 0:00:02
----- 1.8/8.1 MB 5.5 MB/s eta 0:00:02
----- 2.9/8.1 MB 5.2 MB/s eta 0:00:02
----- 4.5/8.1 MB 5.7 MB/s eta 0:00:01
----- 6.0/8.1 MB 6.2 MB/s eta 0:00:01
----- 6.8/8.1 MB 5.7 MB/s eta 0:00:01
----- 8.1/8.1 MB 5.8 MB/s eta 0:00:00
  Downloading contourpy-1.3.3-cp313-cp313-win_amd64.whl (226 kB)
  Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
  Downloading fonttools-4.61.1-cp313-cp313-win_amd64.whl (2.3 MB)
----- 0.0/2.3 MB ? eta -:-:--
----- 1.0/2.3 MB 6.0 MB/s eta 0:00:01
----- 1.6/2.3 MB 4.0 MB/s eta 0:00:01
----- 2.3/2.3 MB 4.0 MB/s eta 0:00:00
  Downloading kiwisolver-1.4.9-cp313-cp313-win_amd64.whl (73 kB)
  Downloading pillow-12.0.0-cp313-cp313-win_amd64.whl (7.0 MB)
----- 0.0/7.0 MB ? eta -:-:--
----- 1.3/7.0 MB 6.8 MB/s eta 0:00:01
----- 2.9/7.0 MB 7.2 MB/s eta 0:00:01
----- 4.7/7.0 MB 7.6 MB/s eta 0:00:01
----- 6.0/7.0 MB 7.5 MB/s eta 0:00:01
----- 7.0/7.0 MB 7.2 MB/s eta 0:00:00
  Downloading pyparsing-3.3.1-py3-none-any.whl (121 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, con-
matplotlib

----- 0/7 [pyparsing]
----- 0/7 [pyparsing]
----- 0/7 [pyparsing]
----- 1/7 [pillow]
----- 1/7 [pillow]
----- 1/7 [pillow]
```



```
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.61.1 kiwisolver-1.4.1  
matplotlib-3.10.8 pillow-12.0.0 pyparsing-3.3.1
```

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 25.1.1 → 25.3

```
In [3]: import matplotlib.pyplot as plt
```

```
In [14]: # 0. Setup & imports ---  
import warnings  
warnings.filterwarnings("ignore")  
  
import os  
from pathlib import Path  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold  
from sklearn.pipeline import Pipeline  
from sklearn.compose import ColumnTransformer  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import StandardScaler, OneHotEncoder  
from sklearn.linear_model import LogisticRegression  
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier  
from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1_score,  
                             roc_auc_score, classification_report, confusion_matrix,  
                             roc_curve, auc, precision_recall_curve)  
import joblib  
  
RND = 42
```

```
OUT_DIR = Path("/mnt/data")
OUT_DIR.mkdir(parents=True, exist_ok=True)
```

```
In [19]: # 1. Load dataset ---
# Change the path if needed
DATA_PATH = Path("C:/Users/hp/OneDrive/Desktop/loan_approval_dataset")
if not DATA_PATH.exists():
    DATA_PATH = Path("loan_approval_dataset.csv")
if not DATA_PATH.exists():
    raise FileNotFoundError(f"Dataset not found at {DATA_PATH} - upload or change the path")

df = pd.read_csv(DATA_PATH)
print("Loaded dataset shape:", df.shape)
display(df.head())
```

Loaded dataset shape: (4269, 13)

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term
0	1	2	Graduate	No	9600000	29900000	12
1	2	0	Not Graduate	Yes	4100000	12200000	8
2	3	3	Graduate	No	9100000	29700000	20
3	4	3	Graduate	No	8200000	30700000	8
4	5	5	Not Graduate	Yes	9800000	24200000	20

```
In [20]: # 2. Clean column names & quick info ---
# Normalize column names - strip spaces, Lower, replace spaces with underscores
df.columns = [str(c).strip().lower().replace(" ", "_") for c in df.columns]
print("Columns:", df.columns.tolist())
print("\nData info:")
display(df.info())

Columns: ['loan_id', 'no_of_dependents', 'education', 'self_employed', 'income_annum',
          'loan_amount', 'loan_term', 'cibil_score', 'residential_assets_value',
          'commercial_assets_value', 'luxury_assets_value', 'bank_asset_value', 'loan_status']

Data info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4269 entries, 0 to 4268
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   loan_id          4269 non-null   int64  
 1   no_of_dependents 4269 non-null   int64  
 2   education        4269 non-null   object  
 3   self_employed    4269 non-null   object  
 4   income_annum     4269 non-null   int64  
 5   loan_amount      4269 non-null   int64  
 6   loan_term        4269 non-null   int64  
 7   cibil_score      4269 non-null   int64  
 8   residential_assets_value 4269 non-null   int64  
 9   commercial_assets_value 4269 non-null   int64  
 10  luxury_assets_value 4269 non-null   int64  
 11  bank_asset_value 4269 non-null   int64  
 12  loan_status      4269 non-null   object 
```

dtypes: int64(10), object(3)
memory usage: 433.7+ KB

None

```
In [21]: # 3. Identify target column robustly and map to 0/1 ---
# Common target names to search
candidates = [c for c in df.columns if c in (
    "loan_status", "status", "approved", "approval", "loan_decision", "loan_appr
if not candidates:
    candidates = [c for c in df.columns if "loan" in c and "status" in c]
if candidates:
    target_col = candidates[0]
else:
    # fallback to last column
    target_col = df.columns[-1]

print(f"Selected target column: '{target_col}'")
print("Unique target values (sample):", df[target_col].unique()[:20])

# Map many common variants to binary
y_raw = df[target_col].astype(str).str.strip().str.lower()
def map_to_binary(s):
    if pd.isna(s):
        return np.nan
    if s in {"y", "yes", "approved", "accept", "accepted", "1", "true", "t"} or
        return 1
    if s in {"n", "no", "rejected", "reject", "0", "false", "f", "deny", "denied"
        return 0
    # fallback: if looks numeric 0/1
    try:
        v = float(s)
        if v == 1.0:
            return 1
        if v == 0.0:
            return 0
    except Exception:
        pass
    # default: try contains 'yes' or 'approve'
    if "yes" in s or "approve" in s:
        return 1
    return 0

y = y_raw.apply(map_to_binary)
print("\nTarget distribution after mapping:")
display(y.value_counts(dropna=False))

# Drop rows where target is missing (NaN)
na_mask = y.isna()
if na_mask.sum() > 0:
    print(f"Dropping {na_mask.sum()} rows with missing target")
    df = df.loc[~na_mask].reset_index(drop=True)
    y = y.loc[~na_mask].reset_index(drop=True)
else:
    y = y.reset_index(drop=True)
```

```
Selected target column: 'loan_status'
Unique target values (sample): [' Approved' ' Rejected']
```

```
Target distribution after mapping:
```

```
loan_status
1    2656
0    1613
Name: count, dtype: int64
```

```
In [22]: # 4. Quick EDA & visualizations (robust/fixed) ---
from IPython.display import display

# Missing values
missing = df.isnull().sum().sort_values(ascending=False)
print("\nColumns with missing values:")
display(missing[missing > 0])

missing_nonzero = missing[missing > 0] # only columns that actually have missing values

if len(missing_nonzero) > 0:
    plt.figure(figsize=(10,4))
    missing_nonzero.plot(kind="bar")
    plt.title("Missing values per column")
    plt.ylabel("Count")
    plt.tight_layout()
    plt.show()
else:
    print("✓ No missing values found in any column. (Skipping missing-values bar.)")

# show class balance (safe-guard if y empty)
if y is not None and len(y) > 0:
    plt.figure(figsize=(5,4))
    # ensure y is a Series (works if y is a numpy array too)
    sns.countplot(x=pd.Series(y))
    plt.title("Target distribution (0=Reject, 1=Approve)")
    plt.xlabel("Loan status")
    plt.ylabel("Count")
    plt.tight_layout()
    plt.show()
else:
    print("Warning: target series 'y' is empty. Skipping class balance plot.")

# numeric vs categorical split (after removing obvious ID columns)
id_like = [c for c in df.columns if c.endswith("_id") or c == "loan_id" or c.startswith("customer_")]
print("ID-like columns (will be dropped):", id_like)

numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
numeric_cols = [c for c in numeric_cols if c not in id_like and c != getattr(y, "name")]
categorical_cols = df.select_dtypes(include=['object', 'category', 'bool']).columns.tolist()
categorical_cols = [c for c in categorical_cols if c != target_col]

print("\nNumeric cols:", numeric_cols)
print("Categorical cols:", categorical_cols)

# histograms for numeric features (guarded)
if numeric_cols:
    try:
        df[numeric_cols].hist(bins=25, figsize=(14, 6))
        plt.suptitle("Numeric feature distributions")
        plt.tight_layout(rect=[0, 0, 1, 0.95])
        plt.show()
    except Exception as e:
        print("Could not plot numeric histograms:", e)
else:
```

```

print("No numeric columns to plot histograms for.")

# top categories for categorical features (bar plots, guarded)
if categorical_cols:
    for c in categorical_cols:
        plt.figure(figsize=(6,3))
        vc = df[c].fillna("MISSING").value_counts().nlargest(10)
        if vc.shape[0] > 0:
            vc.plot(kind="bar")
            plt.title(f"Top categories: {c}")
            plt.tight_layout()
            plt.show()
        else:
            print(f"No categories to plot for column: {c}")
else:
    print("No categorical columns to plot.")

# correlation heatmap for numeric features (including mapped target) - guarded
if numeric_cols:
    try:
        corr_df = df[numeric_cols].copy()
        # add target as numeric for correlation if not present
        corr_df["__target__"] = pd.Series(y).astype(float).values
        corr = corr_df.corr()
        if corr.shape[0] > 0 and corr.shape[1] > 0:
            plt.figure(figsize=(10,8))
            sns.heatmap(corr, annot=False, cmap="coolwarm", center=0)
            plt.title("Correlation matrix (numeric features + target)")
            plt.tight_layout()
            plt.show()
        else:
            print("Correlation matrix is empty; skipping heatmap.")
    except Exception as e:
        print("Could not compute/plot correlation heatmap:", e)
else:
    print("No numeric columns available for correlation heatmap.")

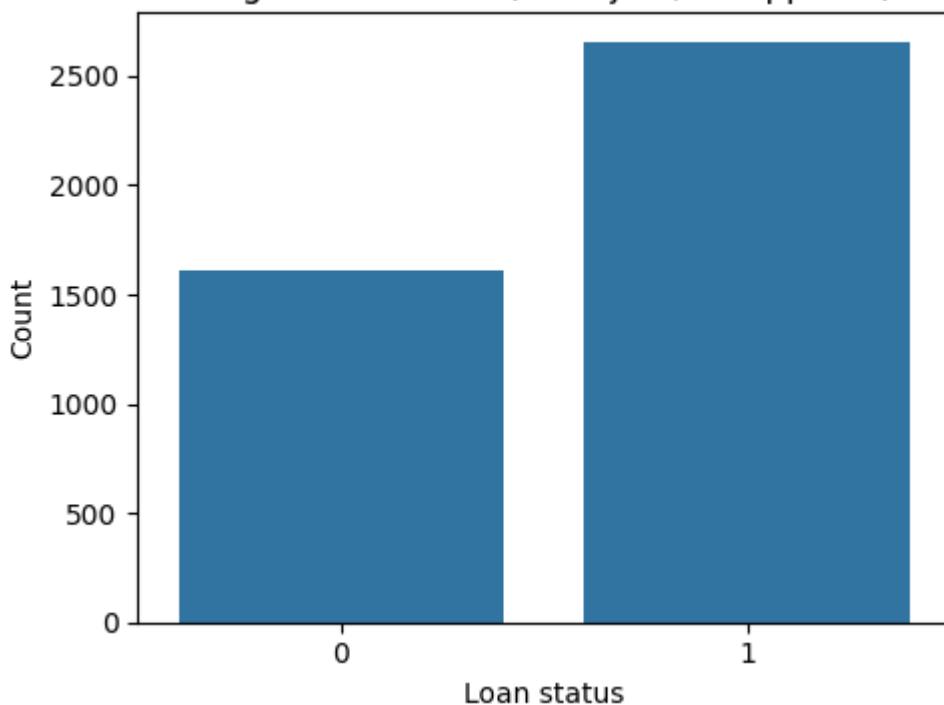
```

Columns with missing values:

Series([], dtype: int64)

No missing values found in any column. (Skipping missing-values bar plot.)

Target distribution (0=Reject, 1=Approve)

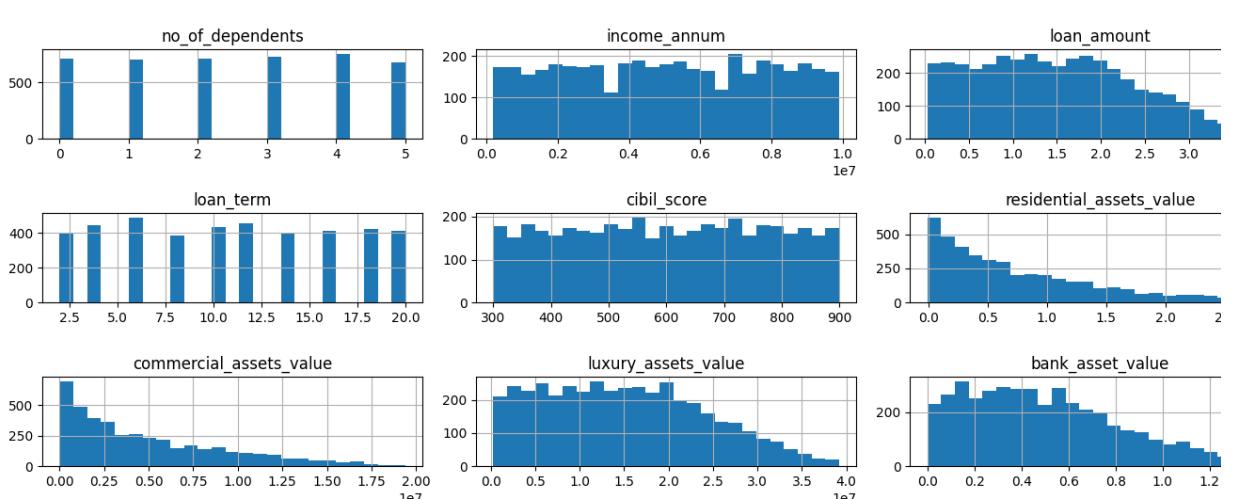


ID-like columns (will be dropped): ['loan_id']

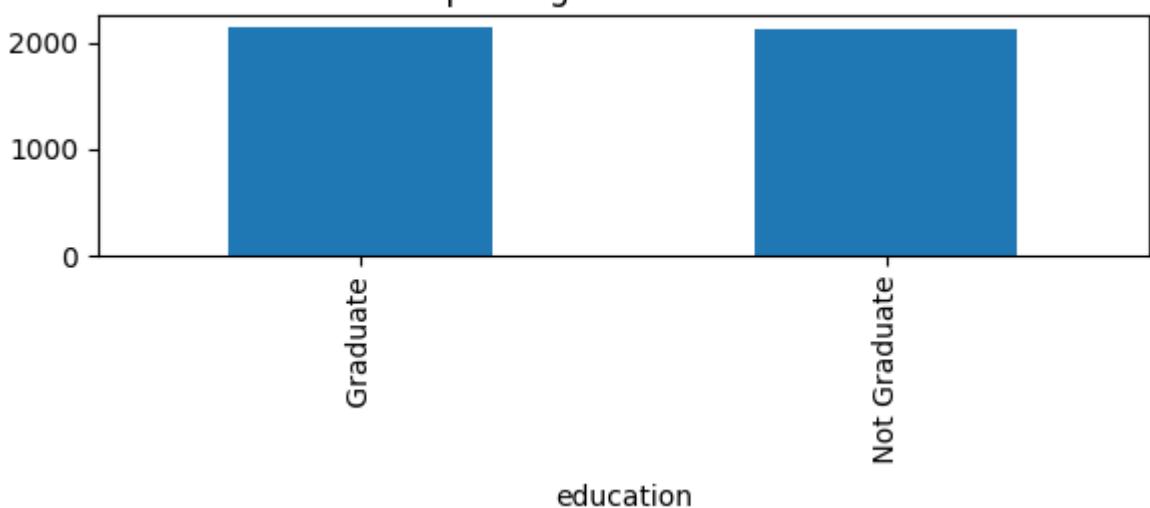
Numeric cols: ['no_of_dependents', 'income_annum', 'loan_amount', 'loan_term',
'cibil_score', 'residential_assets_value', 'commercial_assets_value',
'luxury_assets_value', 'bank_asset_value']

Categorical cols: ['education', 'self_employed']

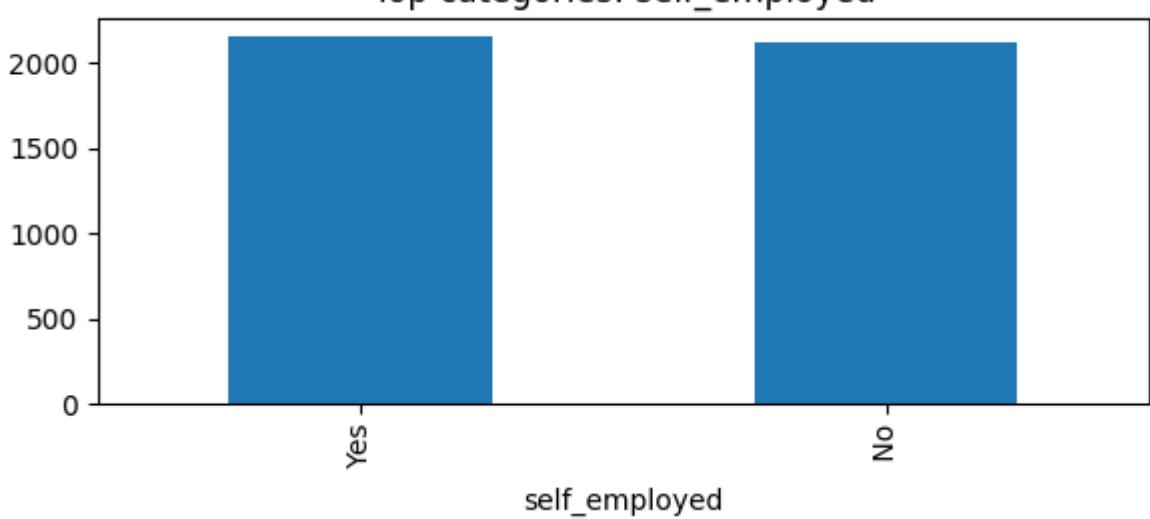
Numeric feature distributions

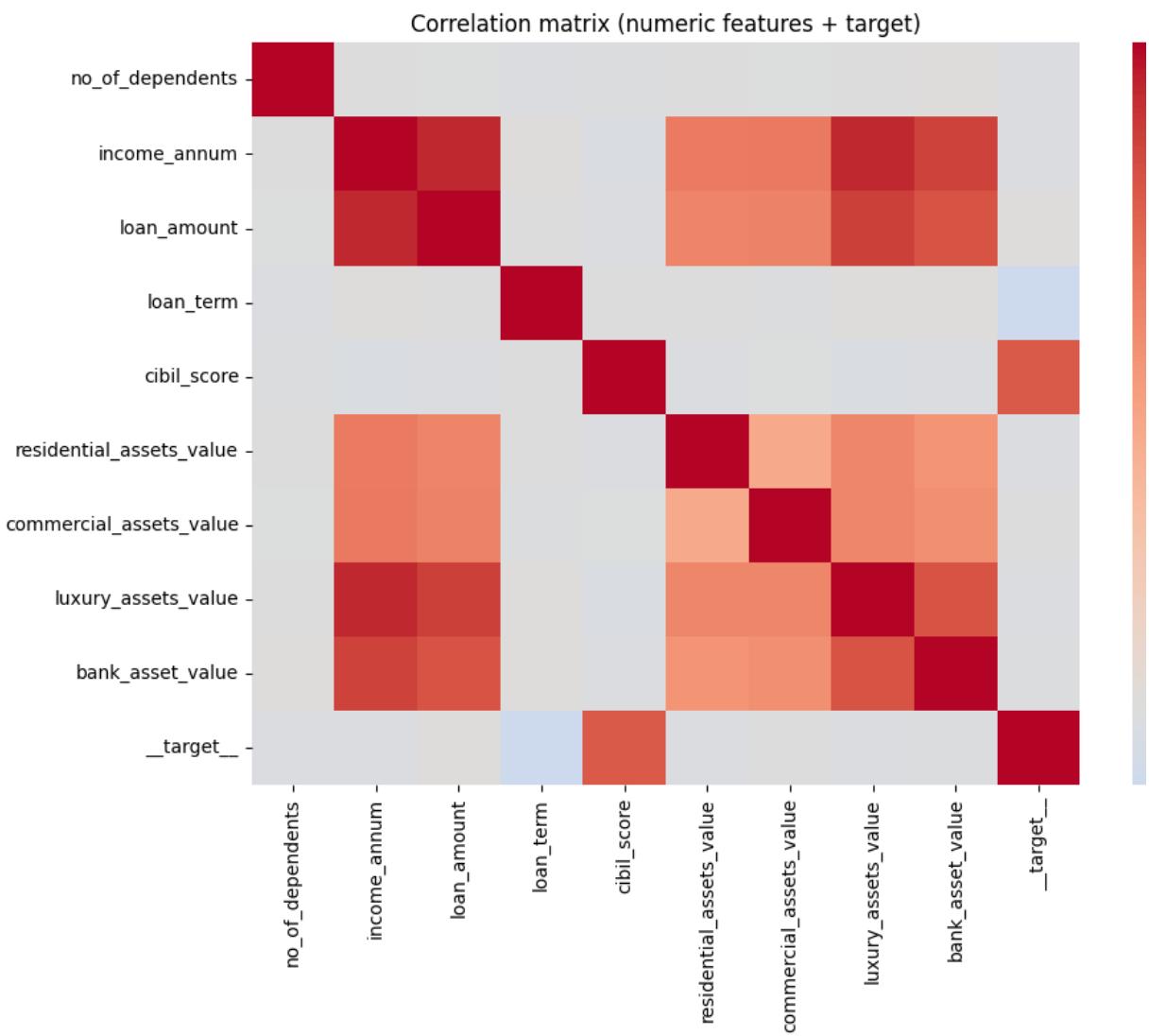


Top categories: education



Top categories: self_employed





In [25]:

```
# 5 Data Split
X = df.drop(columns=[target_col], errors='ignore')
y = pd.Series(y).astype(int)

# Drop ID columns if present
for c in id_like:
    if c in X.columns:
        X = X.drop(columns=c)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

In [27]:

```
# 5. Preprocessing
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline

# Identify numeric and categorical columns
num_cols = X.select_dtypes(include=np.number).columns
cat_cols = X.select_dtypes(include=['object', 'category', 'bool']).columns

# Numeric data pipeline
numeric_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
```

```

        ('scaler', StandardScaler()))
    ])

# Categorical data pipeline (fixed 'sparse' -> 'sparse_output')
categorical_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

# Combine both into a column transformer
preprocessor = ColumnTransformer([
    ('num', numeric_pipe, num_cols),
    ('cat', categorical_pipe, cat_cols)
])

```

In [29]: # 6. Model Training

```

models = {
    "Logistic Regression": LogisticRegression(max_iter=1000, class_weight='balanced'),
    "Random Forest": RandomForestClassifier(n_estimators=200, random_state=42, class_weight='balanced'),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42)
}

results = []

for name, model in models.items():
    pipe = Pipeline([('prep', preprocessor), ('model', model)])
    pipe.fit(X_train, y_train)
    preds = pipe.predict(X_test)
    proba = pipe.predict_proba(X_test)[:, 1] if hasattr(pipe, "predict_proba") else None

    print(f"\n[{name}] Results:")
    print(classification_report(y_test, preds))
    sns.heatmap(confusion_matrix(y_test, preds), annot=True, fmt="d", cmap="Blues")
    plt.title(f"Confusion Matrix - {name}")
    plt.show()

    if proba is not None:
        fpr, tpr, _ = roc_curve(y_test, proba)
        plt.plot(fpr, tpr, label=f"{name} (AUC={auc(fpr,tpr):.2f})")

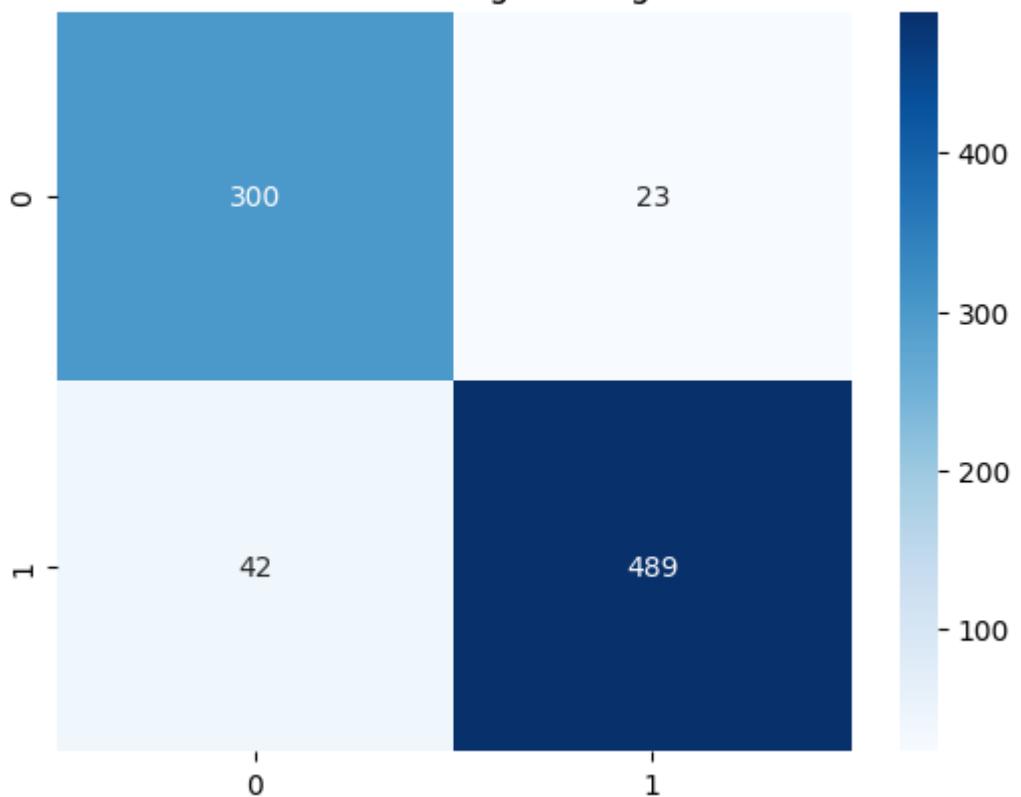
plt.plot([0,1],[0,1], '--', color='gray')
plt.title("ROC Curves")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()

```

[1] Logistic Regression Results:

	precision	recall	f1-score	support
0	0.88	0.93	0.90	323
1	0.96	0.92	0.94	531
accuracy			0.92	854
macro avg	0.92	0.92	0.92	854
weighted avg	0.93	0.92	0.92	854

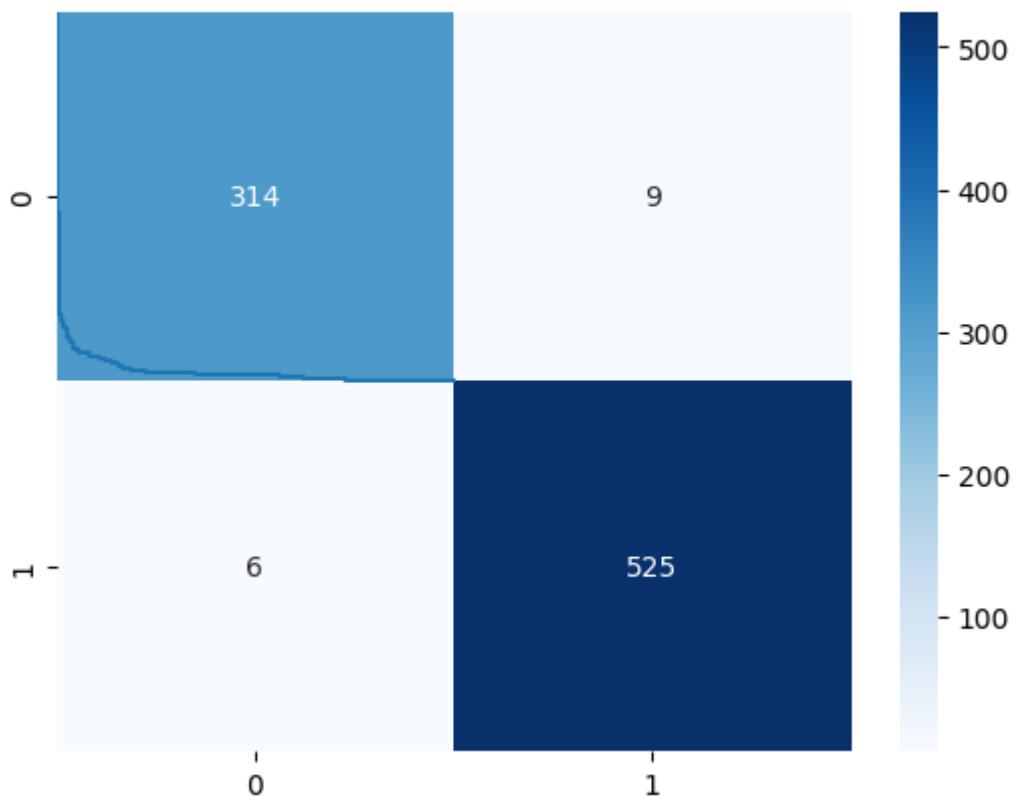
Confusion Matrix - Logistic Regression



Random Forest Results:

	precision	recall	f1-score	support
0	0.98	0.97	0.98	323
1	0.98	0.99	0.99	531
accuracy			0.98	854
macro avg	0.98	0.98	0.98	854
weighted avg	0.98	0.98	0.98	854

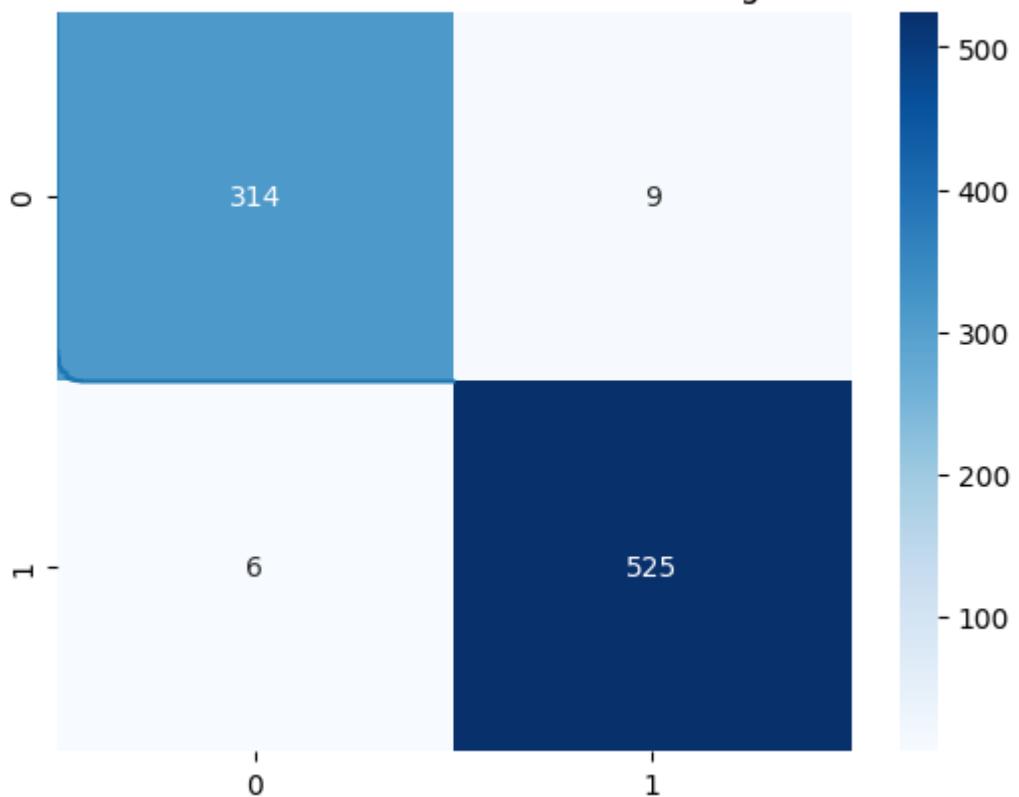
Confusion Matrix - Random Forest



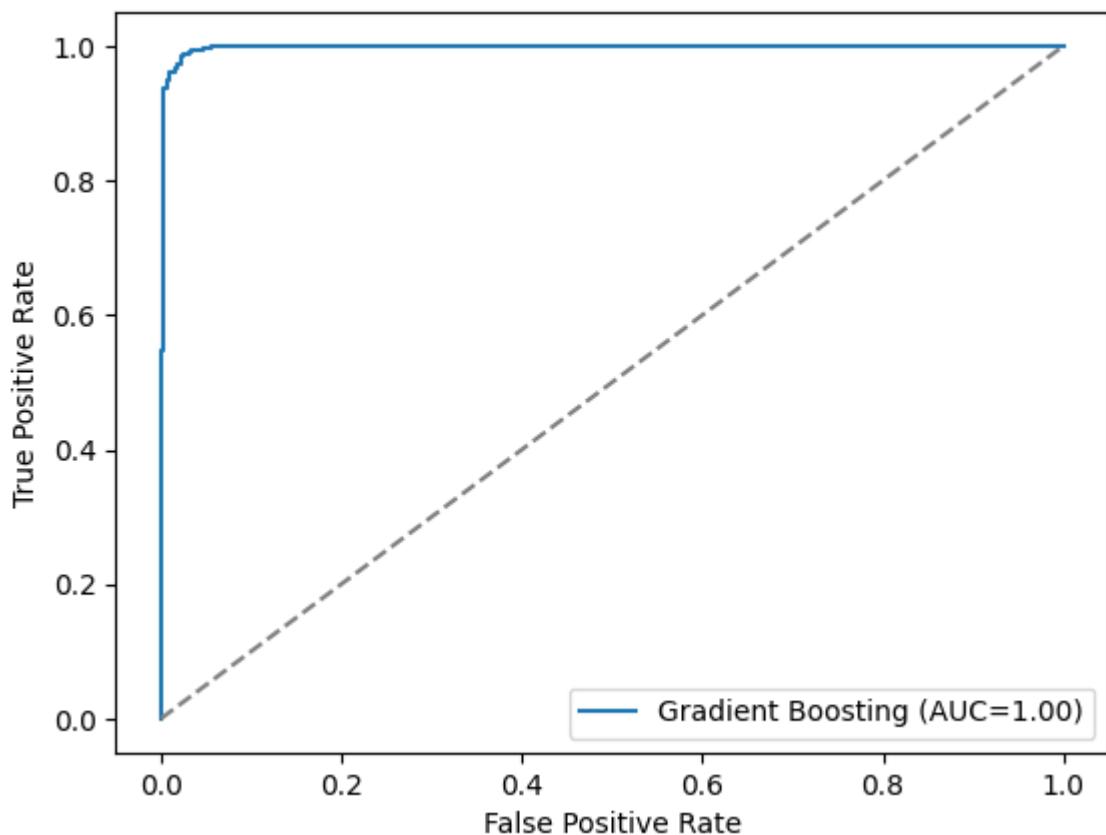
Gradient Boosting Results:

	precision	recall	f1-score	support
0	0.98	0.97	0.98	323
1	0.98	0.99	0.99	531
accuracy			0.98	854
macro avg	0.98	0.98	0.98	854
weighted avg	0.98	0.98	0.98	854

Confusion Matrix - Gradient Boosting



ROC Curves



```
In [31]: # 7. Feature Importance
best_model = models["Random Forest"]
pipe = Pipeline([('prep', preprocessor), ('model', best_model)])
pipe.fit(X_train, y_train)

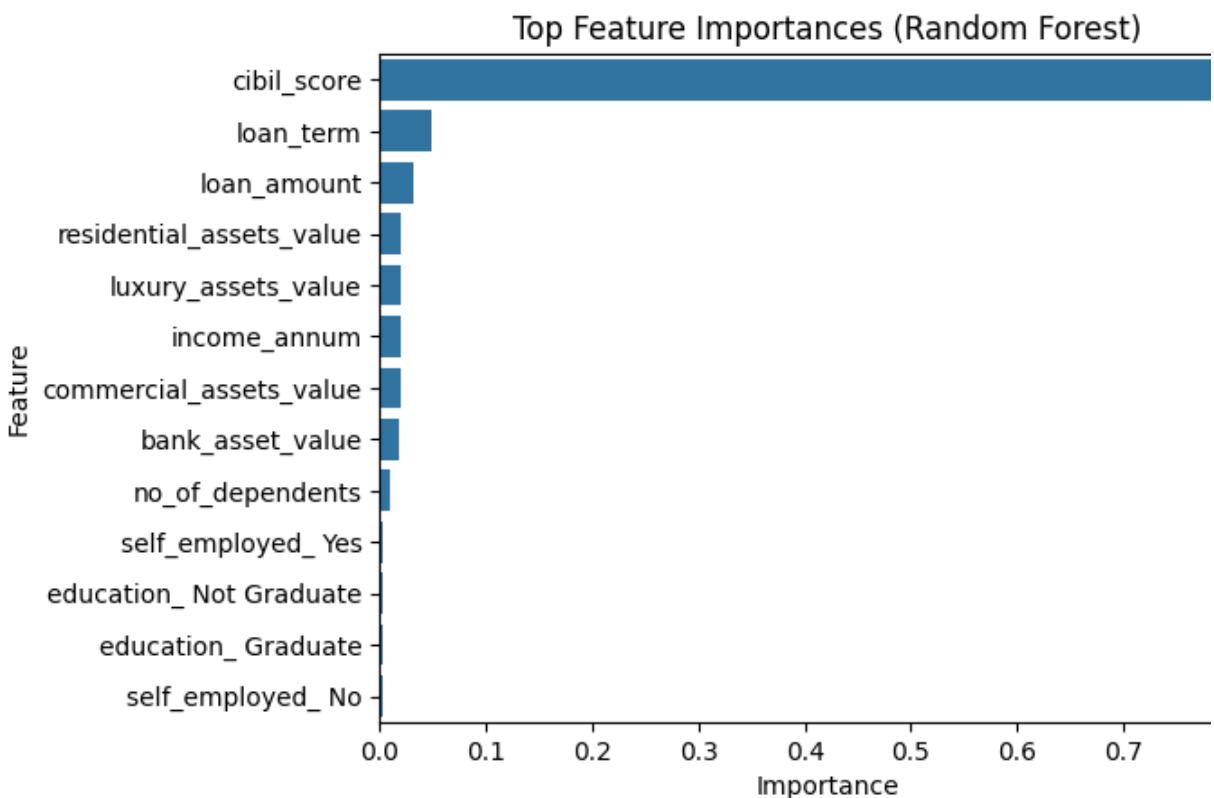
preprocessor.fit(X_train)
```

```

feat_names = list(num_cols) + list(preprocessor.named_transformers_['cat']
                                   .named_steps['onehot'].get_feature_names_out())
importances = pipe.named_steps['model'].feature_importances_
fi = pd.DataFrame({'Feature': feat_names, 'Importance': importances})
fi = fi.sort_values('Importance', ascending=False).head(15)

sns.barplot(x='Importance', y='Feature', data=fi)
plt.title("Top Feature Importances (Random Forest)")
plt.show()

```



In [32]: #8. Save Final Model ---
`joblib.dump(pipe, 'loan_approval_model.pkl')`
`print("✓ Model saved as 'loan_approval_model.pkl'")`

Model saved as 'loan_approval_model.pkl'