Kaggle Playground

Problem Statement / Real World Implementations

1. Importing Libraries

In [1]:
```
# Core Data Science Libraries
import numpy as np
import pandas as pd
import warnings

# Visualization Libraries
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Scikit-learn for Preprocessing and Modeling
from sklearn.model_selection import KFold, train_test_split
from sklearn.preprocessing import OrdinalEncoder, StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Machine Learning Models
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor

# Hyperparameter Tuning
import optuna

# Notebook settings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
```

2. Loading Dataset

In [2]:
```
# Define file paths
TRAIN_PATH = "/kaggle/input/playground-series-s5e11/train.csv"
TEST_PATH = "/kaggle/input/playground-series-s5e11/test.csv"
SUBMISSION_PATH = "/kaggle/input/playground-series-s5e11/sample_submission.csv"

# Load the datasets into pandas DataFrames
train_df = pd.read_csv(TRAIN_PATH)
test_df = pd.read_csv(TEST_PATH)
submission_df = pd.read_csv(SUBMISSION_PATH)
```

In [3]:
```
print("Train shape:", train_df.shape)
print("Test shape:", test_df.shape)
```

```
Train shape: (593994, 13)
Test shape: (254569, 12)
```

In [4]:
```
df=train_df
df.head(5)
```

Out[4]:

	id	annual_income	debt_to_income_ratio	credit_score	loan_amount	interest_rate	gend
0	0	29367.99	0.084	736	2528.42	13.67	Fema
1	1	22108.02	0.166	636	4593.10	12.92	Ma
2	2	49566.20	0.097	694	17005.15	9.76	Ma
3	3	46858.25	0.065	533	4682.48	16.10	Fema
4	4	25496.70	0.053	665	12184.43	10.21	Ma

In [5]:
```
print(df["gender"].unique())
print(df["marital_status"].unique())
print(df["education_level"].unique())
print(df["employment_status"].unique())
print(df["loan_purpose"].unique())
print(df["grade_subgrade"].unique())
```

```
['Female' 'Male' 'Other']
['Single' 'Married' 'Divorced' 'Widowed']
['High School' "Master's" "Bachelor's" 'PhD' 'Other']
['Self-employed' 'Employed' 'Unemployed' 'Retired' 'Student']
['Other' 'Debt consolidation' 'Home' 'Education' 'Vacation' 'Car'
 'Medical' 'Business']
['C3' 'D3' 'C5' 'F1' 'D1' 'D5' 'C2' 'C1' 'F5' 'D4' 'C4' 'D2' 'E5' 'B1'
 'B2' 'F4' 'A4' 'E1' 'F2' 'B4' 'E4' 'B3' 'E3' 'B5' 'E2' 'F3' 'A5' 'A3'
 'A1' 'A2']
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
id                     0
annual_income          0
debt_to_income_ratio   0
credit_score           0
loan_amount            0
interest_rate          0
gender                 0
marital_status         0
education_level        0
employment_status      0
loan_purpose           0
grade_subgrade         0
loan_paid_back         0
dtype: int64
```

In [7]:
```
df.head()
```

Out[7]:

	id	annual_income	debt_to_income_ratio	credit_score	loan_amount	interest_rate	gend
0	0	29367.99	0.084	736	2528.42	13.67	Fema
1	1	22108.02	0.166	636	4593.10	12.92	Ma
2	2	49566.20	0.097	694	17005.15	9.76	Ma
3	3	46858.25	0.065	533	4682.48	16.10	Fema
4	4	25496.70	0.053	665	12184.43	10.21	Ma

4. EDA

In [8]:

```
# Select only numeric columns for correlation matrix
numerical_cols = train_df.select_dtypes(include=np.number).columns.tolist()
numerical_cols.remove('id')
numerical_cols.remove('loan_paid_back')

numeric_df = train_df[numerical_cols + ['loan_paid_back']]
corr_matrix = numeric_df.corr()

# Create the interactive heatmap
fig = go.Figure(data=go.Heatmap(
                    z=corr_matrix.values,
                    x=corr_matrix.columns,
                    y=corr_matrix.columns,
                    colorscale='RdBu_r',
                    zmin=-1, zmax=1,
                    text=corr_matrix.round(2).values,
                    texttemplate="%{text}",
                    hoverongaps=False))

fig.update_layout(
    title='Correlation Heatmap of Numerical Features',
    width=800, height=800
)
fig.show()
```

3. Normalization of data

In [9]:
```
def encode_features(df):
    df_encoded = df.copy()

    # Boolean to integer
    for col in df_encoded.select_dtypes(include='bool').columns:
        df_encoded[col] = df_encoded[col].astype(int)

    # Categorical to integer
    categorical_cols = df_encoded.select_dtypes(include='object').columns
    if len(categorical_cols) > 0:
```

```
        encoder = OrdinalEncoder()
        df_encoded[categorical_cols] = encoder.fit_transform(df_encoded[categori

    return df_encoded

train_ids = train_df['id']
test_ids = test_df['id']

train_processed = encode_features(train_df.drop('id', axis=1))
test_processed = encode_features(test_df.drop('id', axis=1))
```

In [10]:
```
df.head(5)
```

Out[10]:

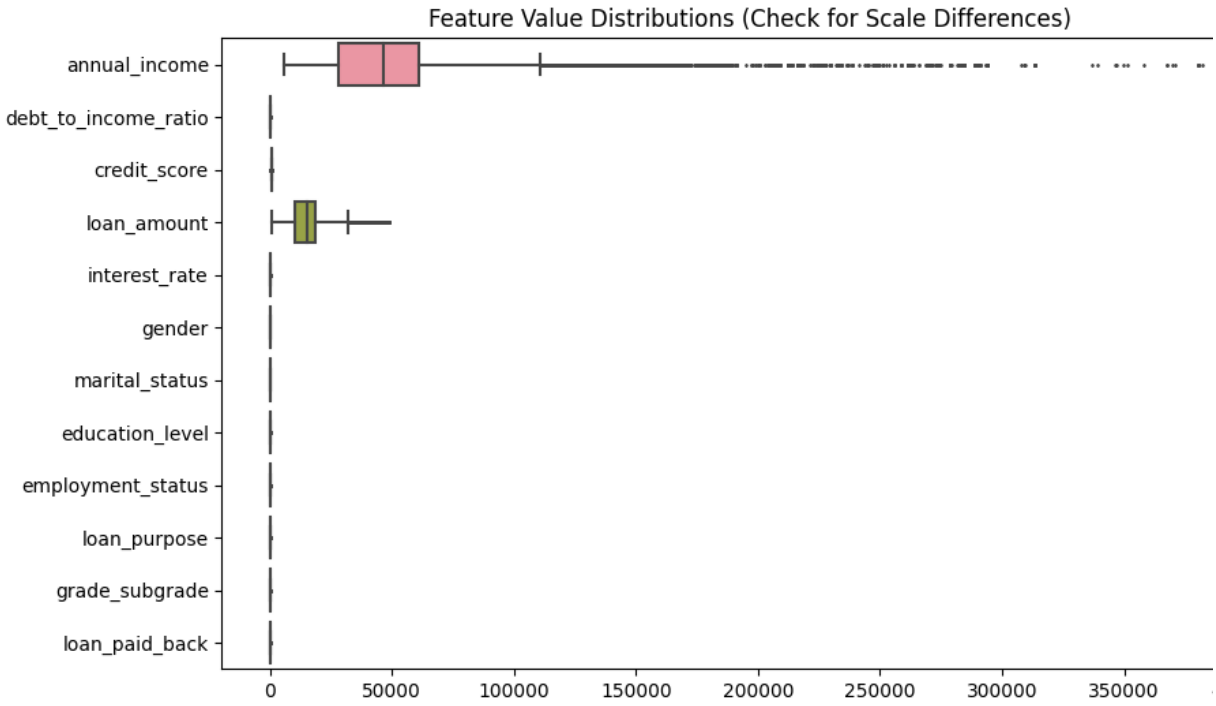
	id	annual_income	debt_to_income_ratio	credit_score	loan_amount	interest_rate	gend
0	0	29367.99	0.084	736	2528.42	13.67	Fema
1	1	22108.02	0.166	636	4593.10	12.92	Ma
2	2	49566.20	0.097	694	17005.15	9.76	Ma
3	3	46858.25	0.065	533	4682.48	16.10	Fema
4	4	25496.70	0.053	665	12184.43	10.21	Ma

In [ ]:
```
# Exclude target column if present
features = train_processed.drop(columns=['loan_paid_back'], errors='ignore')

# 1. Check summary statistics
print("Summary Statistics:\n")
display(features.describe())

# 2. Check for large differences in scale
range_df = features.max() - features.min()
print("\nFeature Ranges:\n")
print(range_df.sort_values(ascending=False))

# 3. Visualize distribution of feature scales
plt.figure(figsize=(10, 6))
sns.boxplot(data=features, orient='h', fliersize=1)
plt.title("Feature Value Distributions (Check for Scale Differences)")
plt.show()

# 4. Correlation check
corr_matrix = features.corr()
high_range_features = range_df[range_df > range_df.mean()].index.tolist()
print(f"\nFeatures with significantly higher ranges: {high_range_features}")

# 5. Quick rule-based decision
if range_df.max() / range_df.min() > 10:
    print("\n✅ Feature scaling is likely necessary (large scale differences det
else:
    print("\n❌ Feature scaling might not be strictly necessary (features on sim
```

```
Summary Statistics:
```

	annual_income	debt_to_income_ratio	credit_score	loan_amount	interest_rate	
count	593994.000000	593994.000000	593994.000000	593994.000000	593994.000000	5
mean	48212.202976	0.120696	680.916009	15020.297629	12.356345	
std	26711.942078	0.068573	55.424956	6926.530568	2.008959	
min	6002.430000	0.011000	395.000000	500.090000	3.200000	
25%	27934.400000	0.072000	646.000000	10279.620000	10.990000	
50%	46557.680000	0.096000	682.000000	15000.220000	12.370000	
75%	60981.320000	0.156000	719.000000	18858.580000	13.680000	
max	393381.740000	0.627000	849.000000	48959.950000	20.990000	

```
Feature Ranges:

annual_income           387379.310
loan_amount              48459.860
credit_score               454.000
grade_subgrade              29.000
interest_rate               17.790
loan_purpose                 7.000
employment_status            4.000
education_level              4.000
marital_status               3.000
gender                       2.000
loan_paid_back               1.000
debt_to_income_ratio         0.616
dtype: float64
```




```
Features with significantly higher ranges: ['annual_income', 'loan_amount']

✅ Feature scaling is likely necessary (large scale differences detected).
```

Train test split

In [12]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler, Pc
import numpy as np

# Use encoded data for model training
X = train_processed.drop("loan_paid_back", axis=1)
y = train_processed["loan_paid_back"]

# Ensure all columns are numeric
X = X.select_dtypes(include=[np.number])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Choose scaling method
selected_method = 'Standard Scaling'

# Apply the best scaling method
if selected_method == 'Min-Max Scaling':
    scaler = MinMaxScaler()
elif selected_method == 'Standard Scaling':
    scaler = StandardScaler()
elif selected_method == 'Robust Scaling':
    scaler = RobustScaler()
elif selected_method == 'Power Transformation':
    scaler = PowerTransformer(method='yeo-johnson')
else:
    scaler = None  # Log or Decimal handled separately

# Perform scaling
if scaler is not None:
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
elif selected_method == 'Log Transformation':
    X_train_scaled = np.log1p(X_train.clip(lower=1e-6))
    X_test_scaled = np.log1p(X_test.clip(lower=1e-6))
elif selected_method == 'Decimal Scaling':
    X_train_scaled = X_train / 100.0
    X_test_scaled = X_test / 100.0
else:
    X_train_scaled = X_train
    X_test_scaled = X_test
```

In [18]:

```
import optuna
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor

# ✅ Enable GPU for both XGBoost and LightGBM

# --- Hyperparameter tuning for XGBRegressor ---
def objective_xgb(trial):
    param = {
        'tree_method': 'gpu_hist',     # Use GPU histogram algorithm
        'predictor': 'gpu_predictor',  # GPU prediction
        'gpu_id': 0,
        'lambda': trial.suggest_loguniform('lambda', 1e-3, 10.0),
```

```
        'alpha': trial.suggest_loguniform('alpha', 1e-3, 10.0),
        'colsample_bytree': trial.suggest_categorical('colsample_bytree', [0.3,
        'subsample': trial.suggest_categorical('subsample', [0.5, 0.6, 0.7, 0.8,
        'learning_rate': trial.suggest_float('learning_rate', 0.005, 0.05, log=T
        'n_estimators': trial.suggest_int('n_estimators', 200, 1000, step=100),
        'max_depth': trial.suggest_int('max_depth', 3, 12),
        'min_child_weight': trial.suggest_int('min_child_weight', 1, 300),
        'random_state': 42
    }

    model = XGBRegressor(**param, verbosity=0)
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    mse = mean_squared_error(y_test, y_pred)
    return mse


# --- Hyperparameter tuning for LGBMRegressor ---
def objective_lgbm(trial):
    param = {
        'device': 'gpu',                # GPU acceleration
        'gpu_platform_id': 0,
        'gpu_device_id': 0,
        'boosting_type': 'gbdt',
        'objective': 'regression',
        'metric': 'mse',

        # Regularization (not too extreme)
        'lambda_l1': trial.suggest_float('lambda_l1', 1e-5, 1.0, log=True),
        'lambda_l2': trial.suggest_float('lambda_l2', 1e-5, 1.0, log=True),

        # Tree and data sampling
        'num_leaves': trial.suggest_int('num_leaves', 16, 256),
        'feature_fraction': trial.suggest_float('feature_fraction', 0.5, 1.0),
        'bagging_fraction': trial.suggest_float('bagging_fraction', 0.5, 1.0),
        'bagging_freq': trial.suggest_int('bagging_freq', 1, 7),
        'min_child_samples': trial.suggest_int('min_child_samples', 10, 100),

        # Learning control
        'learning_rate': trial.suggest_float('learning_rate', 0.005, 0.3, log=Tr
        'n_estimators': trial.suggest_int('n_estimators', 200, 1000, step=100),
        'max_depth': trial.suggest_int('max_depth', 3, 12),

        'random_state': 42,
        'verbosity': -1
    }

    model = LGBMRegressor(**param)
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    mse = mean_squared_error(y_test, y_pred)
    return mse

# --- Run GPU-accelerated Optuna optimization ---
print("🚀 Tuning XGBRegressor (GPU)...")
study_xgb = optuna.create_study(direction='minimize')
study_xgb.optimize(objective_xgb, n_trials=50, timeout=600)
best_params_xgb = study_xgb.best_params
print(f"✅ Best XGBRegressor parameters: {best_params_xgb}")
```

```python
print("\n🚀 Tuning LGBMRegressor (GPU)...")
study_lgbm = optuna.create_study(direction='minimize')
study_lgbm.optimize(objective_lgbm, n_trials=50, timeout=600)
best_params_lgbm = study_lgbm.best_params
print(f"✅ Best LGBMRegressor parameters: {best_params_lgbm}")


# --- Initialize models with tuned GPU parameters ---
xgb_model = XGBRegressor(**best_params_xgb, tree_method='gpu_hist', predictor='g
lgbm_model = LGBMRegressor(**best_params_lgbm, device='gpu')

models = [
    ("XGBRegressor (GPU)", xgb_model),
    ("LGBMRegressor (GPU)", lgbm_model)
]

print("\n⚡ Evaluating Tuned Models on GPU...\n")
mse_scores = []
model_names = []

for name, model in models:
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    mse_scores.append(mse)
    model_names.append(name)

    print(f"{name:<30} | MSE: {mse:.5f} | MAE: {mae:.5f} | R²: {r2:.5f}")
```

```
[I 2025-11-02 17:43:41,777] A new study created in memory with name: no-name-
b985617c-4ea2-418c-92d5-efca381d5437
🚀 Tuning XGBRegressor (GPU)...
```

```
[I 2025-11-02 17:43:46,914] Trial 0 finished with value: 0.07318510225652079 and
parameters: {'lambda': 0.004941577337749069, 'alpha': 1.197400002432866,
'colsample_bytree': 0.5, 'subsample': 0.5, 'learning_rate': 0.018982992354841577,
'n_estimators': 1000, 'max_depth': 5, 'min_child_weight': 231}. Best is trial 0 with
0.07318510225652079.
[I 2025-11-02 17:43:50,003] Trial 1 finished with value: 0.0727613327117026 and paran
{'lambda': 0.49247736085951765, 'alpha': 0.25334079783049696, 'colsample_bytree': 0.7
'subsample': 0.6, 'learning_rate': 0.04486099383361792, 'n_estimators': 600, 'max_dep
5, 'min_child_weight': 282}. Best is trial 1 with value: 0.0727613327117026.
[I 2025-11-02 17:43:53,204] Trial 2 finished with value: 0.07347210397257795 and
parameters: {'lambda': 2.4939774990727757, 'alpha': 0.00889156857409923,
'colsample_bytree': 0.5, 'subsample': 0.5, 'learning_rate': 0.031649457493039185,
'n_estimators': 200, 'max_depth': 11, 'min_child_weight': 184}. Best is trial 1 with
0.0727613327117026.
[I 2025-11-02 17:44:01,883] Trial 3 finished with value: 0.07246169861948995 and
parameters: {'lambda': 0.003243769963262868, 'alpha': 0.10270392875593655,
'colsample_bytree': 0.9, 'subsample': 0.6, 'learning_rate': 0.012735936257358022,
'n_estimators': 800, 'max_depth': 10, 'min_child_weight': 194}. Best is trial 3 with
0.07246169861948995.
[I 2025-11-02 17:44:08,871] Trial 4 finished with value: 0.07242822509145837 and
parameters: {'lambda': 0.0014546625783924889, 'alpha': 0.013812923271668075,
'colsample_bytree': 0.5, 'subsample': 0.6, 'learning_rate': 0.02759269188944516,
'n_estimators': 600, 'max_depth': 10, 'min_child_weight': 108}. Best is trial 4 with
0.07242822509145837.
[I 2025-11-02 17:44:12,439] Trial 5 finished with value: 0.08273784277834692 and
parameters: {'lambda': 2.0318566205991986, 'alpha': 0.6464893804785462, 'colsample_by
0.3, 'subsample': 1.0, 'learning_rate': 0.009708943849169425, 'n_estimators': 400,
'max_depth': 8, 'min_child_weight': 58}. Best is trial 4 with value: 0.0724282250914
[I 2025-11-02 17:44:15,474] Trial 6 finished with value: 0.07280533347069153 and
parameters: {'lambda': 4.412055780689256, 'alpha': 4.929895050593715, 'colsample_bytr
1.0, 'subsample': 0.8, 'learning_rate': 0.028192551277782092, 'n_estimators': 500,
'max_depth': 6, 'min_child_weight': 298}. Best is trial 4 with value: 0.0724282250914
[I 2025-11-02 17:44:20,537] Trial 7 finished with value: 0.0737202811040852 and paran
{'lambda': 6.767365428909071, 'alpha': 1.7354765678588855, 'colsample_bytree': 1.0,
'subsample': 0.8, 'learning_rate': 0.005721484356193055, 'n_estimators': 1000, 'max_d
5, 'min_child_weight': 52}. Best is trial 4 with value: 0.07242822509145837.
[I 2025-11-02 17:44:32,404] Trial 8 finished with value: 0.07260202237911427 and
parameters: {'lambda': 0.044626422582299624, 'alpha': 0.0028675702433960293,
'colsample_bytree': 0.9, 'subsample': 0.8, 'learning_rate': 0.00870731746408398,
'n_estimators': 600, 'max_depth': 12, 'min_child_weight': 102}. Best is trial 4 with
0.07242822509145837.
[I 2025-11-02 17:44:36,015] Trial 9 finished with value: 0.07285908543018464 and
parameters: {'lambda': 0.00451793268375023, 'alpha': 4.393444131293938, 'colsample_by
1.0, 'subsample': 0.6, 'learning_rate': 0.034514101911740196, 'n_estimators': 800,
'max_depth': 4, 'min_child_weight': 130}. Best is trial 4 with value: 0.0724282250914
[I 2025-11-02 17:44:38,576] Trial 10 finished with value: 0.07651479554132842 and
parameters: {'lambda': 0.041688345836198466, 'alpha': 0.026810872230111454,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.01817110688341604,
'n_estimators': 200, 'max_depth': 9, 'min_child_weight': 98}. Best is trial 4 with va
0.07242822509145837.
[I 2025-11-02 17:44:47,397] Trial 11 finished with value: 0.07246750110743767 and
parameters: {'lambda': 0.0010266793541851894, 'alpha': 0.04785571096844009,
'colsample_bytree': 0.9, 'subsample': 0.6, 'learning_rate': 0.012267149180561612,
'n_estimators': 800, 'max_depth': 10, 'min_child_weight': 181}. Best is trial 4 with
0.07242822509145837.
[I 2025-11-02 17:44:57,539] Trial 12 finished with value: 0.07268464775085529 and
parameters: {'lambda': 0.0013696865114221442, 'alpha': 0.18080147018929618,
'colsample_bytree': 0.9, 'subsample': 0.6, 'learning_rate': 0.021623911275339774,
'n_estimators': 800, 'max_depth': 10, 'min_child_weight': 17}. Best is trial 4 with v
0.07242822509145837.
```

```
[I 2025-11-02 17:45:03,330] Trial 13 finished with value: 0.0746030088435929 and
parameters: {'lambda': 0.009764162531166489, 'alpha': 0.0010053073397160099,
'colsample_bytree': 0.3, 'subsample': 0.6, 'learning_rate': 0.013790300084935859,
'n_estimators': 700, 'max_depth': 8, 'min_child_weight': 221}. Best is trial 4 with v
0.07242822509145837.
[I 2025-11-02 17:45:13,092] Trial 14 finished with value: 0.07676782143637838 and
parameters: {'lambda': 0.013123416243427924, 'alpha': 0.012741215180860705,
'colsample_bytree': 0.7, 'subsample': 1.0, 'learning_rate': 0.0059226073096645855,
'n_estimators': 400, 'max_depth': 12, 'min_child_weight': 154}. Best is trial 4 with
0.07242822509145837.
[I 2025-11-02 17:45:22,397] Trial 15 finished with value: 0.07244248944776166 and
parameters: {'lambda': 0.23561745924529676, 'alpha': 0.10330823738380857,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.025603445472810932,
'n_estimators': 900, 'max_depth': 10, 'min_child_weight': 233}. Best is trial 4 with
0.07242822509145837.
[I 2025-11-02 17:45:28,181] Trial 16 finished with value: 0.07247958356599123 and
parameters: {'lambda': 0.29031895533089835, 'alpha': 0.006112492783293742,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.04889888599332788,
'n_estimators': 900, 'max_depth': 7, 'min_child_weight': 259}. Best is trial 4 with v
0.07242822509145837.
[I 2025-11-02 17:45:33,214] Trial 17 finished with value: 0.072491390094319 and paran
{'lambda': 0.17827810746894726, 'alpha': 0.034017131378933856, 'colsample_bytree': 0.
'subsample': 0.7, 'learning_rate': 0.025544181994990197, 'n_estimators': 500, 'max_de
9, 'min_child_weight': 138}. Best is trial 4 with value: 0.07242822509145837.
[I 2025-11-02 17:45:41,602] Trial 18 finished with value: 0.07249989446559847 and
parameters: {'lambda': 0.05765125529052659, 'alpha': 0.06985923709441268,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.034483437148469694,
'n_estimators': 700, 'max_depth': 11, 'min_child_weight': 239}. Best is trial 4 with
0.07242822509145837.
[I 2025-11-02 17:45:50,012] Trial 19 finished with value: 0.07240421300830764 and
parameters: {'lambda': 0.7335732268387569, 'alpha': 0.3889765911702559, 'colsample_by
0.5, 'subsample': 0.7, 'learning_rate': 0.024916587272239006, 'n_estimators': 900,
'max_depth': 9, 'min_child_weight': 90}. Best is trial 19 with value: 0.0724042130083
[I 2025-11-02 17:45:52,867] Trial 20 finished with value: 0.07266643737227771 and
parameters: {'lambda': 1.1366099492736919, 'alpha': 0.46630131239356365,
'colsample_bytree': 0.5, 'subsample': 1.0, 'learning_rate': 0.038591210487037265,
'n_estimators': 400, 'max_depth': 7, 'min_child_weight': 84}. Best is trial 19 with v
0.07240421300830764.
[I 2025-11-02 17:46:01,115] Trial 21 finished with value: 0.07238425248300517 and
parameters: {'lambda': 0.7915769250441695, 'alpha': 0.20295027265648674,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.024618421379043454,
'n_estimators': 900, 'max_depth': 9, 'min_child_weight': 120}. Best is trial 21 with
0.07238425248300517.
[I 2025-11-02 17:46:09,705] Trial 22 finished with value: 0.0723979593769258 and
parameters: {'lambda': 0.7631565009619256, 'alpha': 0.30468250482263903,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.021971445898593338,
'n_estimators': 900, 'max_depth': 9, 'min_child_weight': 70}. Best is trial 21 with v
0.07238425248300517.
[I 2025-11-02 17:46:18,363] Trial 23 finished with value: 0.07238314665826764 and
parameters: {'lambda': 0.7480317106048732, 'alpha': 0.44600429317886986,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.022092181554355675,
'n_estimators': 900, 'max_depth': 9, 'min_child_weight': 60}. Best is trial 23 with v
0.07238314665826764.
[I 2025-11-02 17:46:26,475] Trial 24 finished with value: 0.07237523217815502 and
parameters: {'lambda': 1.318441054025722, 'alpha': 1.246869683296589, 'colsample_bytr
0.7, 'subsample': 0.7, 'learning_rate': 0.016544171654730735, 'n_estimators': 1000,
'max_depth': 8, 'min_child_weight': 23}. Best is trial 24 with value: 0.0723752321781
[I 2025-11-02 17:46:34,816] Trial 25 finished with value: 0.07237474701908335 and
parameters: {'lambda': 2.1545620314964364, 'alpha': 1.5699121002785803, 'colsample_by
0.7, 'subsample': 0.7, 'learning_rate': 0.016554574291763945, 'n_estimators': 1000,
```

```
'max_depth': 8, 'min_child_weight': 4}. Best is trial 25 with value: 0.07237474701908
[I 2025-11-02 17:46:41,676] Trial 26 finished with value: 0.07251275193672542 and
parameters: {'lambda': 9.047128561502541, 'alpha': 2.0727977921513845, 'colsample_byt
0.7, 'subsample': 0.7, 'learning_rate': 0.0156753738304367, 'n_estimators': 1000,
'max_depth': 7, 'min_child_weight': 5}. Best is trial 25 with value: 0.07237474701908
[I 2025-11-02 17:46:49,998] Trial 27 finished with value: 0.07253092780290982 and
parameters: {'lambda': 2.43335368965264, 'alpha': 0.9807314365009324, 'colsample_bytr
0.7, 'subsample': 0.7, 'learning_rate': 0.009647132942360873, 'n_estimators': 1000,
'max_depth': 8, 'min_child_weight': 33}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:46:55,835] Trial 28 finished with value: 0.07285284682343539 and
parameters: {'lambda': 1.5255633460050972, 'alpha': 8.70076153636996, 'colsample_bytr
0.7, 'subsample': 0.5, 'learning_rate': 0.015277979114952087, 'n_estimators': 1000,
'max_depth': 6, 'min_child_weight': 33}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:47:01,547] Trial 29 finished with value: 0.07255461898667484 and
parameters: {'lambda': 3.5960889156904483, 'alpha': 1.0066503250884422, 'colsample_by
0.7, 'subsample': 0.7, 'learning_rate': 0.019960320537031246, 'n_estimators': 1000,
'max_depth': 6, 'min_child_weight': 26}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:47:05,315] Trial 30 finished with value: 0.0747607253873953 and
parameters: {'lambda': 0.11583099523886854, 'alpha': 2.6440802946965194,
'colsample_bytree': 0.7, 'subsample': 0.5, 'learning_rate': 0.011416313040093615,
'n_estimators': 900, 'max_depth': 3, 'min_child_weight': 1}. Best is trial 25 with va
0.07237474701908335.
[I 2025-11-02 17:47:12,689] Trial 31 finished with value: 0.07245805776725403 and
parameters: {'lambda': 0.4431754207132393, 'alpha': 0.6712933892384781, 'colsample_by
0.7, 'subsample': 0.7, 'learning_rate': 0.015533959947632531, 'n_estimators': 900,
'max_depth': 8, 'min_child_weight': 52}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:47:22,286] Trial 32 finished with value: 0.07346675136138331 and
parameters: {'lambda': 0.8457819021436099, 'alpha': 0.1910631637171078, 'colsample_by
0.3, 'subsample': 0.7, 'learning_rate': 0.019827049698242238, 'n_estimators': 1000,
'max_depth': 9, 'min_child_weight': 72}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:47:27,107] Trial 33 finished with value: 0.07249925986826827 and
parameters: {'lambda': 0.41417436503837557, 'alpha': 1.4337242220806041,
'colsample_bytree': 0.7, 'subsample': 0.7, 'learning_rate': 0.022402660552070932,
'n_estimators': 700, 'max_depth': 7, 'min_child_weight': 43}. Best is trial 25 with v
0.07237474701908335.
[I 2025-11-02 17:47:41,675] Trial 34 finished with value: 0.07269116778555898 and
parameters: {'lambda': 1.600043366016401, 'alpha': 0.16871238619580503, 'colsample_by
0.7, 'subsample': 0.7, 'learning_rate': 0.018324646648410194, 'n_estimators': 900,
'max_depth': 11, 'min_child_weight': 18}. Best is trial 25 with value: 0.072374747019
[I 2025-11-02 17:47:48,154] Trial 35 finished with value: 0.07245156458758256 and
parameters: {'lambda': 4.067952082180118, 'alpha': 3.3526624373776137, 'colsample_byt
0.7, 'subsample': 0.7, 'learning_rate': 0.017386740175801 45, 'n_estimators': 800,
'max_depth': 8, 'min_child_weight': 70}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:47:57,166] Trial 36 finished with value: 0.07249549146269255 and
parameters: {'lambda': 0.5887959296091477, 'alpha': 0.6555713587800691, 'colsample_by
0.5, 'subsample': 0.5, 'learning_rate': 0.02933013956533125, 'n_estimators': 1000,
'max_depth': 9, 'min_child_weight': 117}. Best is trial 25 with value: 0.072374747019
[I 2025-11-02 17:48:03,850] Trial 37 finished with value: 0.07423900919185615 and
parameters: {'lambda': 1.100371584329578, 'alpha': 9.665282528458386, 'colsample_bytr
0.3, 'subsample': 0.7, 'learning_rate': 0.013499467486748283, 'n_estimators': 900,
'max_depth': 8, 'min_child_weight': 1}. Best is trial 25 with value: 0.07237474701908
[I 2025-11-02 17:48:14,969] Trial 38 finished with value: 0.07296043682496607 and
parameters: {'lambda': 5.556041197145476, 'alpha': 1.042553115226808, 'colsample_bytr
1.0, 'subsample': 1.0, 'learning_rate': 0.04097351688483964, 'n_estimators': 1000,
'max_depth': 11, 'min_child_weight': 160}. Best is trial 25 with value:
0.07237474701908335.
[I 2025-11-02 17:48:19,734] Trial 39 finished with value: 0.0724933768151354 and
parameters: {'lambda': 3.37747972897794, 'alpha': 0.2767271701785982, 'colsample_bytr
0.5, 'subsample': 0.8, 'learning_rate': 0.0313220085968325, 'n_estimators': 700,
'max_depth': 7, 'min_child_weight': 44}. Best is trial 25 with value: 0.0723747470190
```

```
[I 2025-11-02 17:48:29,107] Trial 40 finished with value: 0.07245685456045288 and
parameters: {'lambda': 0.10768276334372348, 'alpha': 0.11796500323143068,
'colsample_bytree': 0.9, 'subsample': 0.7, 'learning_rate': 0.010917418641055372,
'n_estimators': 800, 'max_depth': 10, 'min_child_weight': 118}. Best is trial 25 with
value: 0.07237474701908335.
[I 2025-11-02 17:48:37,601] Trial 41 finished with value: 0.07240496287588143 and
parameters: {'lambda': 1.7242903565429322, 'alpha': 0.3112942230685287, 'colsample_by
0.5, 'subsample': 0.7, 'learning_rate': 0.02233057634468189, 'n_estimators': 900,
'max_depth': 9, 'min_child_weight': 73}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:48:46,162] Trial 42 finished with value: 0.07240556659475697 and
parameters: {'lambda': 0.3482415603727088, 'alpha': 0.5124663927245346, 'colsample_by
0.5, 'subsample': 0.7, 'learning_rate': 0.022569260128249825, 'n_estimators': 900,
'max_depth': 9, 'min_child_weight': 64}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:48:52,594] Trial 43 finished with value: 0.07255555272624394 and
parameters: {'lambda': 0.8664165158495347, 'alpha': 1.6068925595800625, 'colsample_by
0.5, 'subsample': 0.7, 'learning_rate': 0.01699642166849519, 'n_estimators': 800,
'max_depth': 8, 'min_child_weight': 87}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:49:05,294] Trial 44 finished with value: 0.07281098836825259 and
parameters: {'lambda': 2.370920207675047, 'alpha': 0.7718576119495126, 'colsample_byt
1.0, 'subsample': 0.8, 'learning_rate': 0.020398294253213545, 'n_estimators': 1000,
'max_depth': 10, 'min_child_weight': 19}. Best is trial 25 with value: 0.072374747019
[I 2025-11-02 17:49:14,398] Trial 45 finished with value: 0.07248057438117471 and
parameters: {'lambda': 0.2073135688806521, 'alpha': 0.22681535515972912,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.014135235607927553,
'n_estimators': 900, 'max_depth': 9, 'min_child_weight': 44}. Best is trial 25 with v
0.07237474701908335.
[I 2025-11-02 17:49:20,688] Trial 46 finished with value: 0.07242614768017254 and
parameters: {'lambda': 0.5197056436940581, 'alpha': 0.06463311369302262,
'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.0243676251521161,
'n_estimators': 800, 'max_depth': 8, 'min_child_weight': 106}. Best is trial 25 with
0.07237474701908335.
[I 2025-11-02 17:49:30,302] Trial 47 finished with value: 0.0724615294295614 and
parameters: {'lambda': 1.1645159991466136, 'alpha': 5.957174951962998, 'colsample_byt
0.7, 'subsample': 0.5, 'learning_rate': 0.02817635258933853, 'n_estimators': 1000,
'max_depth': 9, 'min_child_weight': 54}. Best is trial 25 with value: 0.0723747470190
[I 2025-11-02 17:49:41,846] Trial 48 finished with value: 0.07260429009639374 and
parameters: {'lambda': 2.8102328218152093, 'alpha': 0.3685444620484176, 'colsample_by
0.9, 'subsample': 1.0, 'learning_rate': 0.00660693422325644, 'n_estimators': 1000,
'max_depth': 10, 'min_child_weight': 170}. Best is trial 25 with value:
0.07237474701908335.
[I 2025-11-02 17:49:45,448] Trial 49 finished with value: 0.07485800945464388 and
parameters: {'lambda': 9.029441797603612, 'alpha': 0.13288696545420853, 'colsample_by
0.3, 'subsample': 0.6, 'learning_rate': 0.01736632341972285, 'n_estimators': 600,
'max_depth': 6, 'min_child_weight': 135}. Best is trial 25 with value: 0.072374747019
[I 2025-11-02 17:49:45,450] A new study created in memory with name: no-
name-09d26f10-3e62-4992-b552-80fb885350e9
✅ Best XGBRegressor parameters: {'lambda': 2.1545620314964364, 'alpha':
1.5699121002785803, 'colsample_bytree': 0.7, 'subsample': 0.7, 'learning_rate':
0.016554574291763945, 'n_estimators': 1000, 'max_depth': 8, 'min_child_weight': 4}

🚀 Tuning LGBMRegressor (GPU)...
```

```
[I 2025-11-02 17:50:31,344] Trial 0 finished with value: 0.07253775328240906 and
parameters: {'lambda_l1': 0.3144325664664906, 'lambda_l2': 5.139371636614318e-05,
'num_leaves': 68, 'feature_fraction': 0.7739670527825383, 'bagging_fraction':
0.6862650310734147, 'bagging_freq': 2, 'min_child_samples': 66, 'learning_rate':
0.005988000115095095, 'n_estimators': 1000, 'max_depth': 12}. Best is trial 0 with v
0.07253775328240906.
[I 2025-11-02 17:50:50,547] Trial 1 finished with value: 0.0722585602299009 and paran
{'lambda_l1': 0.0009214351567539018, 'lambda_l2': 0.14593609117274167, 'num_leaves':
'feature_fraction': 0.6914298149861955, 'bagging_fraction': 0.5476616408259014,
'bagging_freq': 7, 'min_child_samples': 91, 'learning_rate': 0.05534732687852607,
'n_estimators': 800, 'max_depth': 5}. Best is trial 1 with value: 0.0722585602299009.
[I 2025-11-02 17:51:07,332] Trial 2 finished with value: 0.0749761409277329 and paran
{'lambda_l1': 0.02062168069707945, 'lambda_l2': 0.42134417328030305, 'num_leaves': 21
'feature_fraction': 0.7079408350332993, 'bagging_fraction': 0.6812621030632433,
'bagging_freq': 3, 'min_child_samples': 25, 'learning_rate': 0.008180717678669036,
'n_estimators': 800, 'max_depth': 3}. Best is trial 1 with value: 0.0722585602299009.
[I 2025-11-02 17:51:14,151] Trial 3 finished with value: 0.07322128054279185 and
parameters: {'lambda_l1': 0.14656780612095613, 'lambda_l2': 0.0023605966241375125,
'num_leaves': 164, 'feature_fraction': 0.6317181386259803, 'bagging_fraction':
0.7590420174081491, 'bagging_freq': 2, 'min_child_samples': 88, 'learning_rate':
0.046488582376087946, 'n_estimators': 200, 'max_depth': 5}. Best is trial 1 with valu
0.0722585602299009.
[I 2025-11-02 17:51:39,846] Trial 4 finished with value: 0.07755276125650899 and
parameters: {'lambda_l1': 3.366172877107368e-05, 'lambda_l2': 0.8409720406058278,
'num_leaves': 171, 'feature_fraction': 0.5562417417196953, 'bagging_fraction':
0.7568445513072879, 'bagging_freq': 7, 'min_child_samples': 53, 'learning_rate':
0.2406487051698851, 'n_estimators': 600, 'max_depth': 12}. Best is trial 1 with value
0.0722585602299009.
[I 2025-11-02 17:52:23,066] Trial 5 finished with value: 0.07211290223220379 and
parameters: {'lambda_l1': 0.128918425881384, 'lambda_l2': 0.4922041390426447, 'num_le
85, 'feature_fraction': 0.780118909485992, 'bagging_fraction': 0.7010423180636292,
'bagging_freq': 3, 'min_child_samples': 43, 'learning_rate': 0.013762018055231704,
'n_estimators': 1000, 'max_depth': 10}. Best is trial 5 with value: 0.072112902232203
[I 2025-11-02 17:52:53,086] Trial 6 finished with value: 0.07230714438360515 and
parameters: {'lambda_l1': 7.156655605660374e-05, 'lambda_l2': 0.0004825635840662648,
'num_leaves': 81, 'feature_fraction': 0.7836391942048657, 'bagging_fraction':
0.9726105477466748, 'bagging_freq': 7, 'min_child_samples': 23, 'learning_rate':
0.014392755260006983, 'n_estimators': 700, 'max_depth': 11}. Best is trial 5 with val
0.07211290223220379.
[I 2025-11-02 17:53:32,884] Trial 7 finished with value: 0.07490565396882291 and
parameters: {'lambda_l1': 0.1393319416086843, 'lambda_l2': 0.00011489526741940273,
'num_leaves': 180, 'feature_fraction': 0.9731403292341481, 'bagging_fraction':
0.9506965974055588, 'bagging_freq': 3, 'min_child_samples': 63, 'learning_rate':
0.15018813377646256, 'n_estimators': 700, 'max_depth': 10}. Best is trial 5 with valu
0.07211290223220379.
[I 2025-11-02 17:53:41,440] Trial 8 finished with value: 0.07433481762750607 and
parameters: {'lambda_l1': 0.13027628267843547, 'lambda_l2': 0.025322119540873853,
'num_leaves': 177, 'feature_fraction': 0.7950690606406852, 'bagging_fraction':
0.9876671425651458, 'bagging_freq': 4, 'min_child_samples': 39, 'learning_rate':
0.034485160379812835, 'n_estimators': 400, 'max_depth': 3}. Best is trial 5 with valu
0.07211290223220379.
[I 2025-11-02 17:54:02,986] Trial 9 finished with value: 0.07421014443756509 and
parameters: {'lambda_l1': 0.00018312472660735567, 'lambda_l2': 0.07459340869258678,
'num_leaves': 70, 'feature_fraction': 0.9810870629306392, 'bagging_fraction':
0.8338420490862776, 'bagging_freq': 6, 'min_child_samples': 44, 'learning_rate':
0.19724904693555376, 'n_estimators': 700, 'max_depth': 8}. Best is trial 5 with value
0.07211290223220379.
[I 2025-11-02 17:54:25,117] Trial 10 finished with value: 0.07256668457041368 and
parameters: {'lambda_l1': 0.016647756209526163, 'lambda_l2': 0.010241962337318873,
'num_leaves': 26, 'feature_fraction': 0.8705700435337014, 'bagging_fraction':
```

```
0.5015349878174808, 'bagging_freq': 1, 'min_child_samples': 13, 'learning_rate':
0.017750539333787816, 'n_estimators': 1000, 'max_depth': 9}. Best is trial 5 with val
0.07211290223220379.
[I 2025-11-02 17:54:52,498] Trial 11 finished with value: 0.07224830903598137 and
parameters: {'lambda_l1': 0.0008283291608417681, 'lambda_l2': 0.1317314263139018,
'num_leaves': 119, 'feature_fraction': 0.6622038931501802, 'bagging_fraction':
0.5202962237159514, 'bagging_freq': 5, 'min_child_samples': 94, 'learning_rate':
0.07943821465744355, 'n_estimators': 900, 'max_depth': 6}. Best is trial 5 with value
0.07211290223220379.
[I 2025-11-02 17:55:24,566] Trial 12 finished with value: 0.07234643374580316 and
parameters: {'lambda_l1': 0.0018807887617167645, 'lambda_l2': 0.10095735108490883,
'num_leaves': 117, 'feature_fraction': 0.523605280565048, 'bagging_fraction':
0.6127548384124035, 'bagging_freq': 5, 'min_child_samples': 75, 'learning_rate':
0.09155640266635312, 'n_estimators': 1000, 'max_depth': 6}. Best is trial 5 with valu
0.07211290223220379.
[I 2025-11-02 17:56:02,783] Trial 13 finished with value: 0.07202691678367223 and
parameters: {'lambda_l1': 0.008671827328713605, 'lambda_l2': 0.9534495401773614,
'num_leaves': 93, 'feature_fraction': 0.6175025610817721, 'bagging_fraction':
0.5976865256682474, 'bagging_freq': 5, 'min_child_samples': 100, 'learning_rate':
0.022950075585759594, 'n_estimators': 900, 'max_depth': 8}. Best is trial 13 with val
0.07202691678367223.
[I 2025-11-02 17:56:16,096] Trial 14 finished with value: 0.0730424184046366 and
parameters: {'lambda_l1': 0.97796183403284, 'lambda_l2': 0.9892421772014676, 'num_lea
20, 'feature_fraction': 0.5829913402030946, 'bagging_fraction': 0.6149886257283502,
'bagging_freq': 4, 'min_child_samples': 77, 'learning_rate': 0.024130578807453878,
'n_estimators': 500, 'max_depth': 8}. Best is trial 13 with value: 0.0720269167836722
[I 2025-11-02 17:56:55,572] Trial 15 finished with value: 0.07222724244991853 and
parameters: {'lambda_l1': 0.014025358578827501, 'lambda_l2': 1.329058809983528e-05,
'num_leaves': 89, 'feature_fraction': 0.8770715298194027, 'bagging_fraction':
0.8652887096279431, 'bagging_freq': 5, 'min_child_samples': 38, 'learning_rate':
0.012475276302429575, 'n_estimators': 900, 'max_depth': 10}. Best is trial 13 with va
0.07202691678367223.
[I 2025-11-02 17:57:47,465] Trial 16 finished with value: 0.07209149167253599 and
parameters: {'lambda_l1': 0.02984918018920208, 'lambda_l2': 0.0038900141278855136,
'num_leaves': 253, 'feature_fraction': 0.6077525016605771, 'bagging_fraction':
0.6238771661713605, 'bagging_freq': 3, 'min_child_samples': 47, 'learning_rate':
0.027635238542258328, 'n_estimators': 900, 'max_depth': 9}. Best is trial 13 with val
0.07202691678367223.
[I 2025-11-02 17:58:19,309] Trial 17 finished with value: 0.07212572174075227 and
parameters: {'lambda_l1': 0.007002129712547195, 'lambda_l2': 0.0023849684437820913,
'num_leaves': 211, 'feature_fraction': 0.6085888418432391, 'bagging_fraction':
0.5875211476096043, 'bagging_freq': 4, 'min_child_samples': 57, 'learning_rate':
0.026356134760624365, 'n_estimators': 800, 'max_depth': 7}. Best is trial 13 with val
0.07202691678367223.
[I 2025-11-02 17:58:35,967] Trial 18 finished with value: 0.07227692383518781 and
parameters: {'lambda_l1': 0.03824144030552098, 'lambda_l2': 0.0005319149118124057,
'num_leaves': 252, 'feature_fraction': 0.5059218251437705, 'bagging_fraction':
0.6446790822101208, 'bagging_freq': 6, 'min_child_samples': 100, 'learning_rate':
0.02684976287649066, 'n_estimators': 300, 'max_depth': 9}. Best is trial 13 with valu
0.07202691678367223.
[I 2025-11-02 17:59:05,772] Trial 19 finished with value: 0.0723308807111972 and
parameters: {'lambda_l1': 0.0033709477325440866, 'lambda_l2': 0.020340445048691293,
'num_leaves': 250, 'feature_fraction': 0.6504416827519438, 'bagging_fraction':
0.5677678188262886, 'bagging_freq': 1, 'min_child_samples': 77, 'learning_rate':
0.06906257487362563, 'n_estimators': 900, 'max_depth': 7}. Best is trial 13 with valu
0.07202691678367223.
[I 2025-11-02 17:59:35,951] Trial 20 finished with value: 0.07347882454567486 and
parameters: {'lambda_l1': 1.0777188084812856e-05, 'lambda_l2': 0.0008885474587640863,
'num_leaves': 142, 'feature_fraction': 0.7171889047093466, 'bagging_fraction':
0.7981692020410982, 'bagging_freq': 2, 'min_child_samples': 30, 'learning_rate':
```

```
0.12096084246117793, 'n_estimators': 600, 'max_depth': 9}. Best is trial 13 with valu
0.07202691678367223.
[I 2025-11-02 18:00:10,209] Trial 21 finished with value: 0.0723735182059255 and
parameters: {'lambda_l1': 0.053439864656311564, 'lambda_l2': 0.48324670329760644,
'num_leaves': 52, 'feature_fraction': 0.8490067938619557, 'bagging_fraction':
0.6861016894589134, 'bagging_freq': 3, 'min_child_samples': 46, 'learning_rate':
0.011941546951601316, 'n_estimators': 900, 'max_depth': 10}. Best is trial 13 with va
0.07202691678367223.
✅ Best LGBMRegressor parameters: {'lambda_l1': 0.008671827328713605, 'lambda_l2':
0.9534495401773614, 'num_leaves': 93, 'feature_fraction': 0.6175025610817721,
'bagging_fraction': 0.5976865256682474, 'bagging_freq': 5, 'min_child_samples': 100,
'learning_rate': 0.022950075585759594, 'n_estimators': 900, 'max_depth': 8}

⚡ Evaluating Tuned Models on GPU...

XGBRegressor (GPU)              | MSE: 0.07237 | MAE: 0.14805 | R²: 0.55105
LGBMRegressor (GPU)             | MSE: 0.07190 | MAE: 0.14756 | R²: 0.55395
```

In [19]:
```python
# Select best model
best_idx = np.argmin(mse_scores)
best_model_name = model_names[best_idx]
best_model = models[best_idx][1]
print(f"\n✅ Best Model Based on MSE: {best_model_name}")
```

```
✅ Best Model Based on MSE: LGBMRegressor (GPU)
```

In [20]:
```python
# Evaluate final model
y_pred = best_model.predict(X_test_scaled)
mse_default = mean_squared_error(y_test, y_pred)
mae_default = mean_absolute_error(y_test, y_pred)
r2_default = r2_score(y_test, y_pred)

print("\n✅ Final Model Evaluation:")
print(f"Mean Squared Error : {mse_default:.5f}")
print(f"Mean Absolute Error: {mae_default:.5f}")
print(f"R² Score : {r2_default:.5f}")
```

```
✅ Final Model Evaluation:
Mean Squared Error : 0.07190
Mean Absolute Error: 0.14756
R² Score : 0.55395
```

Selecting best model and Generating Submission

In [22]:
```python
print("\n🚀 Retraining the best model on full training data...")

# Prepare full training features and target
X_full = train_processed.drop(columns=['loan_paid_back'], errors='ignore')
y_full = train_processed['loan_paid_back']

# Ensure all columns are numeric
X_full = X_full.select_dtypes(include=[np.number])

# Scale full data using the same scaler
if scaler is not None:
    X_full_scaled = scaler.fit_transform(X_full)
else:
    X_full_scaled = X_full

# Retrain best model on the full scaled dataset
```

```
best_model.fit(X_full_scaled, y_full)

print(f"✅ Model retrained successfully: {best_model_name} ({best_model.__class_
```

```
🚀 Retraining the best model on full training data...
✅ Model retrained successfully: LGBMRegressor
```

In [23]:
```
# Keep IDs for submission if available
if 'id' in test_df.columns:
    test_ids = test_df['id']
else:
    test_ids = range(len(test_df))  # create sequential IDs if missing

# Encode test data (using your encode_features function)
test_processed = encode_features(test_df.drop('id', axis=1, errors='ignore'))

# Ensure numeric columns only
X_submission = test_processed.select_dtypes(include=[np.number])

# Scale using the same scaler
if scaler is not None:
    X_submission_scaled = scaler.transform(X_submission)
else:
    X_submission_scaled = X_submission
```

In [24]:
```
print("🔮 Generating predictions using the best model...")
submission_preds = best_model.predict(X_submission_scaled)

# Optional: clip predictions to valid range [0, 1]
submission_preds = np.clip(submission_preds, 0, 1)
```

```
🔮 Generating predictions using the best model...
```

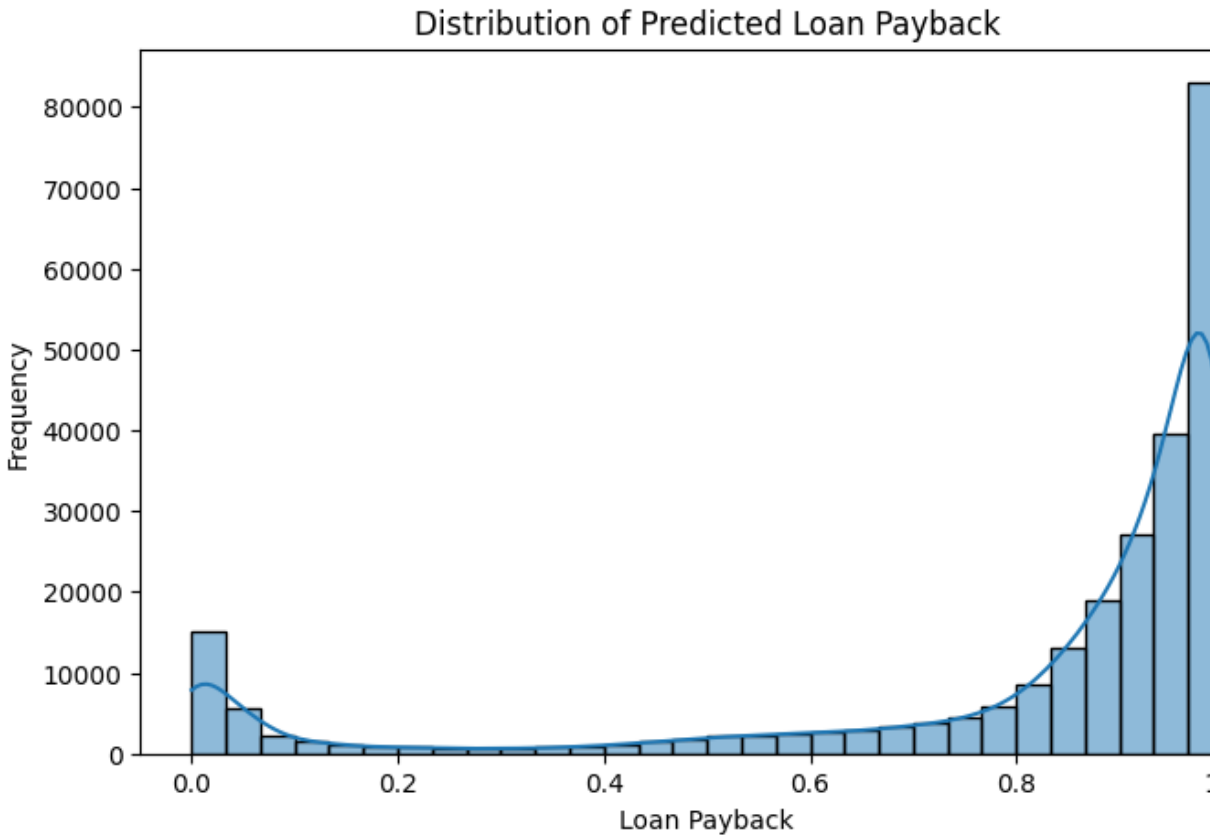
In [25]:
```
submission = pd.DataFrame({
    'id': test_ids,
    'loan_paid_back': submission_preds
})
submission.to_csv('submission.csv', index=False)
print("\n✅ Submission file 'submission.csv' generated successfully!")
display(submission.head())
```

```
✅ Submission file 'submission.csv' generated successfully!
```

	id	loan_paid_back
0	593994	0.961028
1	593995	0.981874
2	593996	0.489422
3	593997	0.927867
4	593998	0.969554

In [26]:
```
plt.figure(figsize=(8, 5))
sns.histplot(submission['loan_paid_back'], bins=30, kde=True)
plt.title('Distribution of Predicted Loan Payback')
plt.xlabel('Loan Payback')
```

```
plt.ylabel('Frequency')
plt.show()
```



In [ ]: