# Week 2 Notebook – Predicting Early Mortgage Delinquency

## Peer Review Note

Hi peers,

This notebook walks through my Week 2 exploration of the Home Credit dataset as part of my capstone project on predicting early mortgage delinquency.

My approach includes: - **Exploratory Data Analysis (EDA):** I examined distributions, class imbalance, and engineered a new feature (INCOME\_CREDIT\_RATIO) to capture borrower leverage. - **Multicollinearity Check:** I used Variance Inflation Factor (VIF) to detect and address feature redundancy. Based on high VIF values, I dropped AMT\_GOODS\_PRICE. - **Leakage Check:** I scanned for post-loan features that could bias the model and retained only application-time variables. - **Modeling:** I applied Lasso, Ridge, and Elastic Net regression using cross-validation. All models were evaluated using R² and RMSE.

### Key Insight:

The models produced low R² scores (~0.006), indicating weak linear relationships in the current feature set. While performance was limited, this regression work establishes a baseline and confirms the need for nonlinear modeling and classification methods in the next phase.

**Dataset:** [Home Credit Default Risk Dataset on Kaggle](https://www.kaggle.com/competitions/home-credit-default-risk/data)

I welcome any feedback on: - Feature engineering opportunities - Ways to improve EDA flow or visuals - Alternative regularization/tuning suggestions - Any other suggestions/ideas

Thanks in advance for your review!

# 2.2 Activity: Week 2 Jupyter Notebook – Linear Regression 2

Assignment: Week 2 Jupyter Notebook — Linear Regression 2 Each week, you will apply the concepts of that week to your Integrated Capstone Project’s dataset. In preparation for Milestone One, create a Jupyter Notebook (similar to in Module B, semester two) that illustrates these lessons. There are no specific questions to answer in your Jupyter Notebook files in this course; your general goal is to analyze your data, using the methods you have learned about in this course and in this program, and draw interesting conclusions.

For Week 2, include concepts such as linear regression with lasso, ridge, and elastic net regression. This homework will be submitted for peer review and feedback in Week 3 in the assignment titled 3.4 Peer Review: Week 2 Jupyter Notebook.

import matplotlib.pyplot as plt  
import seaborn as sns  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import StandardScaler  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LassoCV, RidgeCV, ElasticNetCV  
from sklearn.metrics import r2\_score, mean\_squared\_error  
import numpy as np

df = pd.read\_csv("application\_train.csv")

#structure  
#review top 5 rows  
df.head()

SK\_ID\_CURR

TARGET

NAME\_CONTRACT\_TYPE

CODE\_GENDER

FLAG\_OWN\_CAR

FLAG\_OWN\_REALTY

CNT\_CHILDREN

AMT\_INCOME\_TOTAL

AMT\_CREDIT

AMT\_ANNUITY

…

FLAG\_DOCUMENT\_18

FLAG\_DOCUMENT\_19

FLAG\_DOCUMENT\_20

FLAG\_DOCUMENT\_21

AMT\_REQ\_CREDIT\_BUREAU\_HOUR

AMT\_REQ\_CREDIT\_BUREAU\_DAY

AMT\_REQ\_CREDIT\_BUREAU\_WEEK

AMT\_REQ\_CREDIT\_BUREAU\_MON

AMT\_REQ\_CREDIT\_BUREAU\_QRT

AMT\_REQ\_CREDIT\_BUREAU\_YEAR

0

100002

1

Cash loans

M

N

Y

0

202500.0

406597.5

24700.5

…

0

0

0

0

0.0

0.0

0.0

0.0

0.0

1.0

1

100003

0

Cash loans

F

N

N

0

270000.0

1293502.5

35698.5

…

0

0

0

0

0.0

0.0

0.0

0.0

0.0

0.0

2

100004

0

Revolving loans

M

Y

Y

0

67500.0

135000.0

6750.0

…

0

0

0

0

0.0

0.0

0.0

0.0

0.0

0.0

3

100006

0

Cash loans

F

N

Y

0

135000.0

312682.5

29686.5

…

0

0

0

0

NaN

NaN

NaN

NaN

NaN

NaN

4

100007

0

Cash loans

M

N

Y

0

121500.0

513000.0

21865.5

…

0

0

0

0

0.0

0.0

0.0

0.0

0.0

0.0

5 rows × 122 columns

#review number of rows and columns  
df.shape

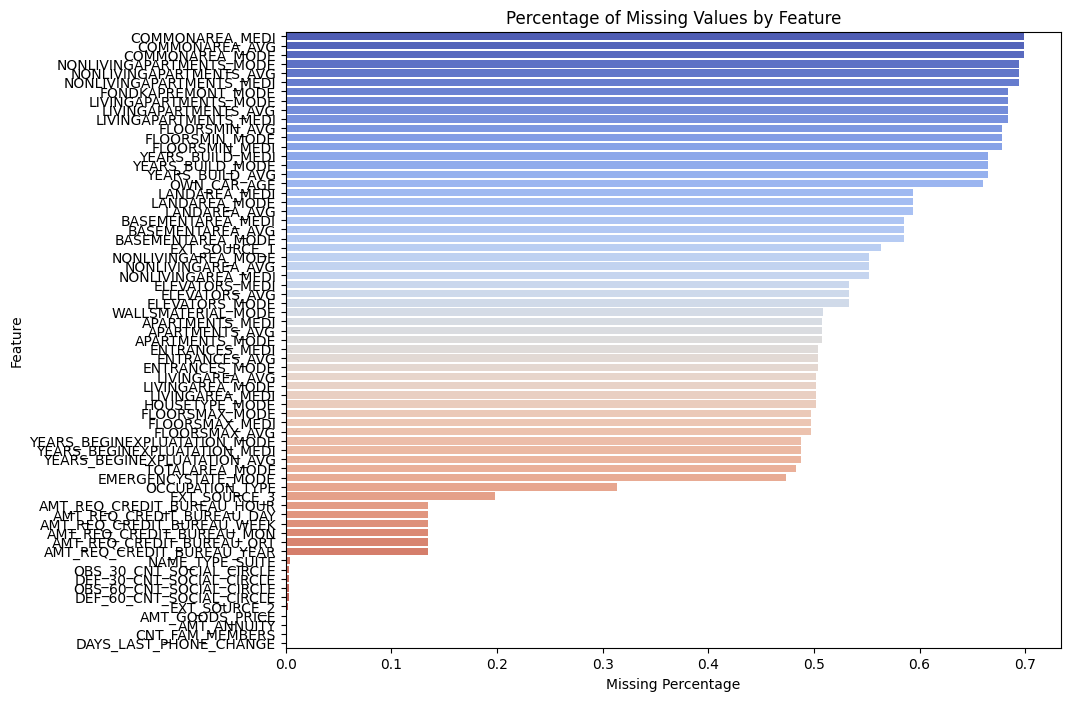
(307511, 122)

df.describe

<bound method NDFrame.describe of SK\_ID\_CURR TARGET NAME\_CONTRACT\_TYPE CODE\_GENDER FLAG\_OWN\_CAR \  
0 100002 1 Cash loans M N   
1 100003 0 Cash loans F N   
2 100004 0 Revolving loans M Y   
3 100006 0 Cash loans F N   
4 100007 0 Cash loans M N   
... ... ... ... ... ...   
307506 456251 0 Cash loans M N   
307507 456252 0 Cash loans F N   
307508 456253 0 Cash loans F N   
307509 456254 1 Cash loans F N   
307510 456255 0 Cash loans F N   
  
 FLAG\_OWN\_REALTY CNT\_CHILDREN AMT\_INCOME\_TOTAL AMT\_CREDIT \  
0 Y 0 202500.0 406597.5   
1 N 0 270000.0 1293502.5   
2 Y 0 67500.0 135000.0   
3 Y 0 135000.0 312682.5   
4 Y 0 121500.0 513000.0   
... ... ... ... ...   
307506 N 0 157500.0 254700.0   
307507 Y 0 72000.0 269550.0   
307508 Y 0 153000.0 677664.0   
307509 Y 0 171000.0 370107.0   
307510 N 0 157500.0 675000.0   
  
 AMT\_ANNUITY ... FLAG\_DOCUMENT\_18 FLAG\_DOCUMENT\_19 FLAG\_DOCUMENT\_20 \  
0 24700.5 ... 0 0 0   
1 35698.5 ... 0 0 0   
2 6750.0 ... 0 0 0   
3 29686.5 ... 0 0 0   
4 21865.5 ... 0 0 0   
... ... ... ... ... ...   
307506 27558.0 ... 0 0 0   
307507 12001.5 ... 0 0 0   
307508 29979.0 ... 0 0 0   
307509 20205.0 ... 0 0 0   
307510 49117.5 ... 0 0 0   
  
 FLAG\_DOCUMENT\_21 AMT\_REQ\_CREDIT\_BUREAU\_HOUR AMT\_REQ\_CREDIT\_BUREAU\_DAY \  
0 0 0.0 0.0   
1 0 0.0 0.0   
2 0 0.0 0.0   
3 0 NaN NaN   
4 0 0.0 0.0   
... ... ... ...   
307506 0 NaN NaN   
307507 0 NaN NaN   
307508 0 1.0 0.0   
307509 0 0.0 0.0   
307510 0 0.0 0.0   
  
 AMT\_REQ\_CREDIT\_BUREAU\_WEEK AMT\_REQ\_CREDIT\_BUREAU\_MON \  
0 0.0 0.0   
1 0.0 0.0   
2 0.0 0.0   
3 NaN NaN   
4 0.0 0.0   
... ... ...   
307506 NaN NaN   
307507 NaN NaN   
307508 0.0 1.0   
307509 0.0 0.0   
307510 0.0 2.0   
  
 AMT\_REQ\_CREDIT\_BUREAU\_QRT AMT\_REQ\_CREDIT\_BUREAU\_YEAR   
0 0.0 1.0   
1 0.0 0.0   
2 0.0 0.0   
3 NaN NaN   
4 0.0 0.0   
... ... ...   
307506 NaN NaN   
307507 NaN NaN   
307508 0.0 1.0   
307509 0.0 0.0   
307510 0.0 1.0   
  
[307511 rows x 122 columns]>

#calc % missing for each column  
missing\_pct = df.isnull().mean().sort\_values(ascending=False)  
  
#filter to columns with missing data  
missing\_pct = missing\_pct[missing\_pct > 0]  
  
plt.figure(figsize=(10, 8))  
sns.barplot(x=missing\_pct.values, y=missing\_pct.index, palette="coolwarm")  
plt.title("Percentage of Missing Values by Feature")  
plt.xlabel("Missing Percentage")  
plt.ylabel("Feature")  
plt.show()

/var/folders/2q/3t5h8lp97hz63t941xlydf4w0000gn/T/ipykernel\_77202/2148702148.py:8: FutureWarning:   
  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.  
  
 sns.barplot(x=missing\_pct.values, y=missing\_pct.index, palette="coolwarm")



png

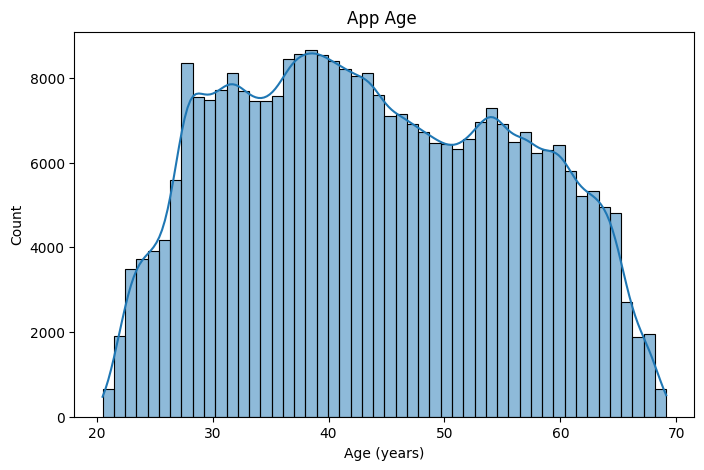
#class imbalance review   
#if more than ~80% of cases are one class, may need to review sampling   
default\_counts = df['TARGET'].value\_counts()  
default\_percent = df['TARGET'].value\_counts(normalize=True) \* 100  
  
print("Default counts:\n", default\_counts)  
print("\nDefault percentages:\n", default\_percent)

Default counts:  
 TARGET  
0 282686  
1 24825  
Name: count, dtype: int64  
  
Default percentages:  
 TARGET  
0 91.927118  
1 8.072882  
Name: proportion, dtype: float64

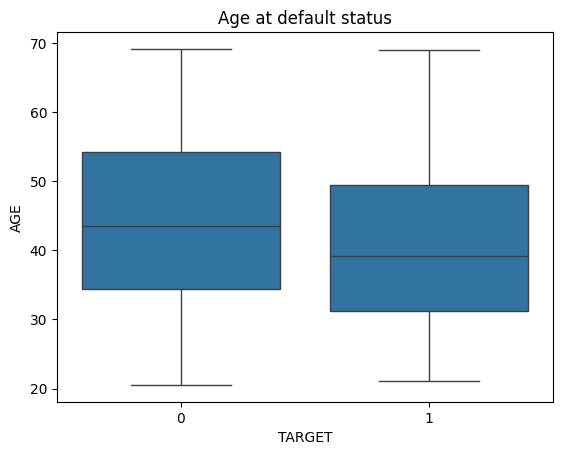
Analysis of default count –

* 91.9% of applicants did not default
* Only 8.1% of applicants did default
* This is a highly imbalanced dataset. This should be OK since it is common in credit risk modeling – most applicants do not default. We will need to adjust for imbalance.

#convert DAYS\_BIRTH to age in years  
# needed to convert to age because of how the data was structured in the file.  
# the raw variable DAYS\_BIRTH represented a negative number of days before the loan application, such as -12,000 days.  
df['AGE'] = (-df['DAYS\_BIRTH']) / 365  
  
plt.figure(figsize=(8, 5))  
sns.histplot(df['AGE'], bins=50, kde=True)  
plt.title("App Age ")  
plt.xlabel("Age (years)")  
plt.show()  
  
sns.boxplot(x='TARGET', y='AGE', data=df)  
plt.title("Age at default status")  
plt.show()



png

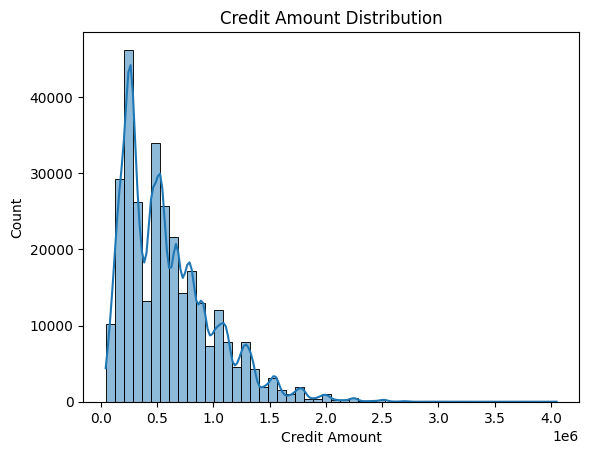


png

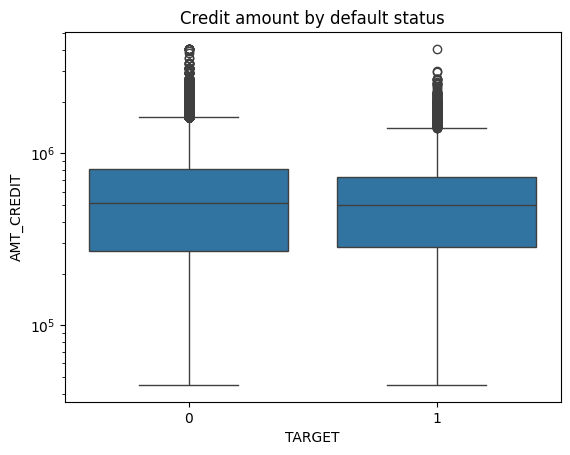
Analysis of plts – 1. Histogram / age - Most applicants are between 25 and 60 years old. - There’s a peak in the 30s and 40s, with a gradual decline afterward. - very few applicants are over 65 or under 25. - supports that the dataset covers working-age adults – which makes sense for loan applications.

1. Boxplot / age vs default Status
   * Apps who defaulted (TARGET = 1) have a slightly lower median age than those who repaid.
   * The interquartile range (middle 50%) of defaulters is slightly lower than non-defaulters.
   * Both groups span a similar full range (roughly 20–70), but defaulters skew younger.
   * borrowers are slightly more likely to default, suggesting that age may be a useful predictive feature.

#analyze credit amount   
#help flag risk factors and outliers tied to borrower age, loan size, etc.  
sns.histplot(df['AMT\_CREDIT'], bins=50, kde=True)  
plt.title("Credit Amount Distribution")  
plt.xlabel("Credit Amount")  
plt.show()  
  
sns.boxplot(x='TARGET', y='AMT\_CREDIT', data=df)  
plt.yscale('log')  
plt.title("Credit amount by default status")  
plt.show()

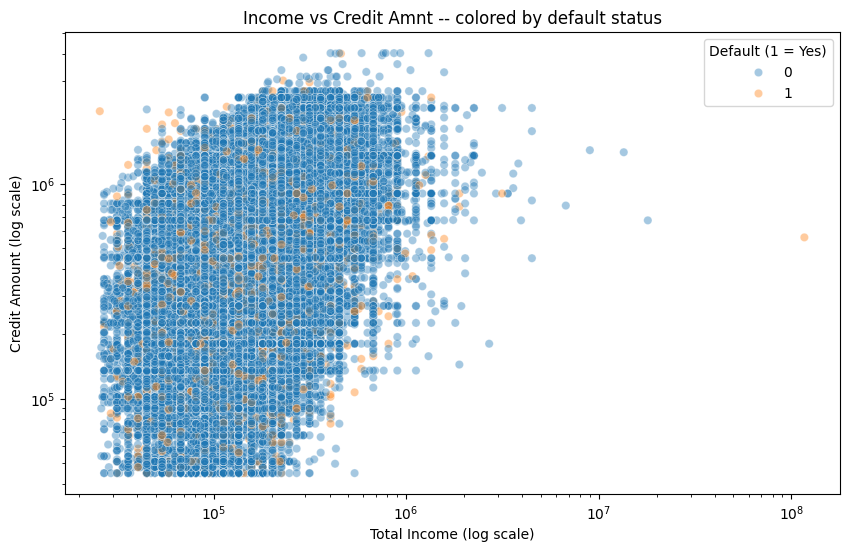


png



png

#income vs. credit amount by defalt   
#view relationship between total income and credit amount   
# and how that relationship differs between defaulters and non-defaulters  
plt.figure(figsize=(10, 6))  
  
sns.scatterplot(  
 data=df,  
 x='AMT\_INCOME\_TOTAL',  
 y='AMT\_CREDIT',  
 hue='TARGET',  
 alpha=0.4  
)  
  
plt.xscale('log')  
plt.yscale('log')  
plt.title("Income vs Credit Amnt -- colored by default status")  
plt.xlabel("Total Income (log scale)")  
plt.ylabel("Credit Amount (log scale)")  
plt.legend(title='Default (1 = Yes)')  
plt.show()



png

Analysis of plot above – Income vs Credit Amnt – colored by default

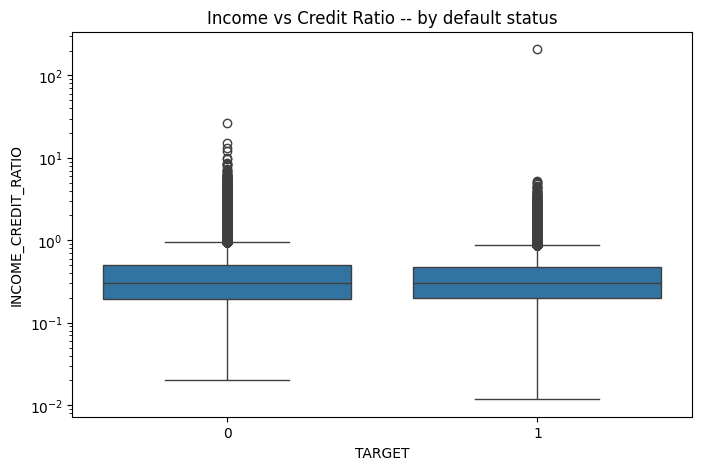
1.most appls have income between $50,000 – $5000,000, and credit amount between $100,000 – $1,000,000

Majority of points are blue (TARGET = 0 = non-default), which is expected given the earlier imbalance (only ~8% defaulted which aligns with our previous findings).

Orange points (TARGET = 1 = default) appear more frequently in the lower-income / high-credit region, suggesting some highher risk.

Due to this, I will create a plot showiung the income to credit ratio below. The idea is that a low ratio of income to credit may mean that someone is overleveraged.

#income to credit amount plot  
df['INCOME\_CREDIT\_RATIO'] = df['AMT\_INCOME\_TOTAL'] / df['AMT\_CREDIT']  
plt.figure(figsize=(8, 5))  
sns.boxplot(x='TARGET', y='INCOME\_CREDIT\_RATIO', data=df)  
plt.yscale('log')  
plt.title("Income vs Credit Ratio -- by default status")  
plt.show()



png

Analysis if plot above – Income vs Credit Ratio – by default status

* Both groups (default = 1, non-default = 0) have a similar median ratio.
* defaulters (TARGET = 1) show slightly lower typical ratios on average and tighter interquartile range (middle 50%)
* a few outliers existr in both groups, but not many.

#Data leakage check  
#performed a keyword-based scan for common leakage indicators such as "PAYMENT", "DRAWING", "LATE", "OVERDUE", "DAYS\_LAST", and "CNT\_PAYMENT".  
leakage\_keywords = ['PAYMENT', 'DRAWING', 'LATE', 'OVERDUE', 'DAYS\_LAST', 'CNT\_PAYMENT']  
suspicious\_cols = [col for col in df.columns if any(k in col.upper() for k in leakage\_keywords)]  
print("Possible leakage-related columns:")  
print(suspicious\_cols)

Possible leakage-related columns:  
['DAYS\_LAST\_PHONE\_CHANGE']

Anaylsis and outcome of the data leak check

The only feature flagged was DAYS\_LAST\_PHONE\_CHANGE: This represents the number of days since the borrower last changed their phone number. Although it’s included in the application data and likely available at loan approval, it could be subject to updates post-application in some operational systems.

I did not remove DAYS\_LAST\_PHONE\_CHANGE for now, as it does not clearly represent post-loan behavior. Can monitor during model evaluation to ensure it does not influence performance in a skewed way. No other known leakage-prone fields (e.g., actual payment amounts, credit usage behavior) were present in the dataset at this stage.

#CALC VIF  
#select numerical features that are probably relevant and numerical (not category)  
features\_for\_vif = [  
 'AMT\_INCOME\_TOTAL',  
 'AMT\_CREDIT',  
 'AMT\_ANNUITY',  
 'AMT\_GOODS\_PRICE',  
 'AGE', # previously engineered from DAYS\_BIRTH  
 'INCOME\_CREDIT\_RATIO' # engineered feature  
]  
  
#drop rows with missing values in these columns  
df\_vif = df[features\_for\_vif].dropna()

#scale data   
scaler = StandardScaler()  
X\_vif\_scaled = scaler.fit\_transform(df\_vif)

#create VIF dataframe  
vif\_df = pd.DataFrame()  
vif\_df["Feature"] = df\_vif.columns  
vif\_df["VIF"] = [variance\_inflation\_factor(X\_vif\_scaled, i) for i in range(X\_vif\_scaled.shape[1])]  
  
#sort  
vif\_df.sort\_values("VIF", ascending=False)

Feature

VIF

1

AMT\_CREDIT

40.225996

3

AMT\_GOODS\_PRICE

39.685118

5

INCOME\_CREDIT\_RATIO

4.696424

0

AMT\_INCOME\_TOTAL

4.279054

2

AMT\_ANNUITY

2.649128

4

AGE

1.013134

### Multicollinearity Check (VIF)

I calculated the Variance Inflation Factor (VIF) for a subset of key numerical features. Two features AMT\_CREDIT and AMT\_GOODS\_PRICE, which showed VIFs over 39, indicating strong multicollinearity.

I dropped AMT\_GOODS\_PRICE and retained AMT\_CREDIT as it more accurately reflects the actual loan size. See cleaned set below.

All remaining features have VIF < 5, suggesting acceptable multicollinearity levels for linear modeling.

# set the cleaned list of features   
features\_for\_modeling = [  
 'AMT\_INCOME\_TOTAL',  
 'AMT\_CREDIT',  
 'AMT\_ANNUITY',  
 'AGE',  
 'INCOME\_CREDIT\_RATIO'  
]  
  
# create df, drop rows with missing values  
df\_vif = df[features\_for\_modeling].dropna()  
  
#standardize the features   
scaler = StandardScaler()  
X\_vif\_scaled = scaler.fit\_transform(df\_vif)  
  
#calc VIF for each feature  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  
import pandas as pd  
  
vif\_df = pd.DataFrame()  
vif\_df["Feature"] = df\_vif.columns  
vif\_df["VIF"] = [variance\_inflation\_factor(X\_vif\_scaled, i) for i in range(X\_vif\_scaled.shape[1])]  
  
#sorted results  
vif\_df.sort\_values("VIF", ascending=False)

Feature

VIF

4

INCOME\_CREDIT\_RATIO

4.673061

0

AMT\_INCOME\_TOTAL

4.266343

1

AMT\_CREDIT

3.130202

2

AMT\_ANNUITY

2.586815

3

AGE

1.013022

### VIF on cleaned feature set

After identifying strong multicollinearity between AMT\_CREDIT and AMT\_GOODS\_PRICE, I removed AMT\_GOODS\_PRICE and recalculated VIF.

The final selected features include: - AMT\_INCOME\_TOTAL - AMT\_CREDIT - AMT\_ANNUITY - AGE - INCOME\_CREDIT\_RATIO

All VIF scores were under 10, suggesting acceptable multicollinearity levels for linear regression modeling.

### Feature Engineering

During EDA, I created a new feature to reflect borrower leverage:

* INCOME\_CREDIT\_RATIO = AMT\_INCOME\_TOTAL / AMT\_CREDIT

This ratio captures a borrower’s income relative to their requested loan amount, acting as a proxy for repayment capacity. It passed multicollinearity checks and was retained as part of the final feature set used for modeling.

### Next step >> Create modeling pipeline using Lasso Regression (LassoCV), Ridge Regression (RidgeCV), Elastic Net (ElasticNetCV)

##PREP DATA  
#using final/clean dataset with selected features and no missing values  
model\_df = df[features\_for\_modeling + ['TARGET']].dropna()  
X = model\_df[features\_for\_modeling]  
y = model\_df['TARGET']  
  
#split data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify=y, test\_size=0.2, random\_state=42)  
  
#standardize features  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
  
##FIT LASSO MODEL  
lasso = LassoCV(cv=5, random\_state=42)  
lasso.fit(X\_train\_scaled, y\_train)  
lasso\_preds = lasso.predict(X\_test\_scaled)  
  
  
#FIT RIDGE MODEL  
ridge = RidgeCV(alphas=np.logspace(-6, 6, 13), cv=5)  
ridge.fit(X\_train\_scaled, y\_train)  
ridge\_preds = ridge.predict(X\_test\_scaled)  
  
  
##FIT ELASTIC NET MODEL   
elastic = ElasticNetCV(cv=5, random\_state=42)  
elastic.fit(X\_train\_scaled, y\_train)  
elastic\_preds = elastic.predict(X\_test\_scaled)  
  
  
##EVAL MODELS  
def evaluate(y\_true, y\_pred, model\_name):  
 r2 = r2\_score(y\_true, y\_pred)  
 rmse = mean\_squared\_error(y\_true, y\_pred) \*\* 0.5  
 print(f"{model\_name} | R^2: {r2:.4f} | RMSE: {rmse:.4f}")  
  
evaluate(y\_test, lasso\_preds, "Lasso")  
evaluate(y\_test, ridge\_preds, "Ridge")  
evaluate(y\_test, elastic\_preds, "Elastic Net")

Lasso | R^2: 0.0060 | RMSE: 0.2716  
Ridge | R^2: 0.0061 | RMSE: 0.2716  
Elastic Net | R^2: 0.0060 | RMSE: 0.2716

### Model Performance Summary

| Model | R² | RMSE | Notes |
| --- | --- | --- | --- |
| Lasso | 0.0060 | 0.2716 | R² is extremely low — the model explains almost no variance |
| Ridge | 0.0061 | 0.2716 | Nearly identical — very limited predictive power |
| Elastic Net | 0.0060 | 0.2716 | Same story — reinforces the weak linear relationship |

The regularized linear models are barely outperforming a flat line; R² values this low indicate almost no explanatory strength. An RMSE of ~0.27, given a binary (0/1) outcome, suggests the models are mostly predicting the average of the target rather than capturing any real signal. The linear relationships between these features and early delinquency are weak. This could mean: - The current features just aren’t strong predictors - The signal might be nonlinear or based on interactions - The target definition could use refinement, for example, narrowing in on 30–60 day delinquencies instead of lumping all defaults together

### Model Performance / Analysis

I ran Lasso, Ridge, and Elastic Net regressions using a cleaned set of features with low multicollinearity. All three models returned very low R² values (around 0.006) and an RMSE of about 0.271. This suggests that the selected features within a linear framework don’t offer much predictive value for the target:

* There doesn’t seem to be a strong linear relationship between the predictors and early delinquency
* Nonlinear models or interactions might be necessary to capture more signal
* Since the target is binary, a classification approach could be a better fit overall

These results give me a good baseline to compare against more complex models later on.

## Summary & Next Steps

This Week 2 notebook focused on building a foundational understanding of the Home Credit dataset and applying regularized regression techniques (Lasso, Ridge, Elastic Net) to establish a baseline.

### Accomplishments / Completed

* Explored the data: looked at distributions, class imbalance, and some feature relationships
* Created a new feature, INCOME\_CREDIT\_RATIO, to capture borrower leverage
* Handled multicollinearity by checking VIF and dropping highly correlated features
* Trained and evaluated three regularized linear models using cross-validation

### Main Takeaway

The R² scores were very low (around 0.006), which suggests that the features used don’t have strong linear relationships with the target. So, this really confirmed the need to shift toward non-linear models and treat this as a classification problem instead.

### Next Up

* Reframe the problem as binary classification
* Try logistic regression and tree-based models like Random Forest and XGBoost
* Use oversampling (like SMOTE) to help with the class imbalance
* Add and test more engineered features and interactions

This sets me up with a solid base for Milestone 1. I’d love any feedback, especially on feature engineering ideas or model setup improvements!