# Low Level Design (LLD) Credit Card Default Prediction

# Problem Statement:

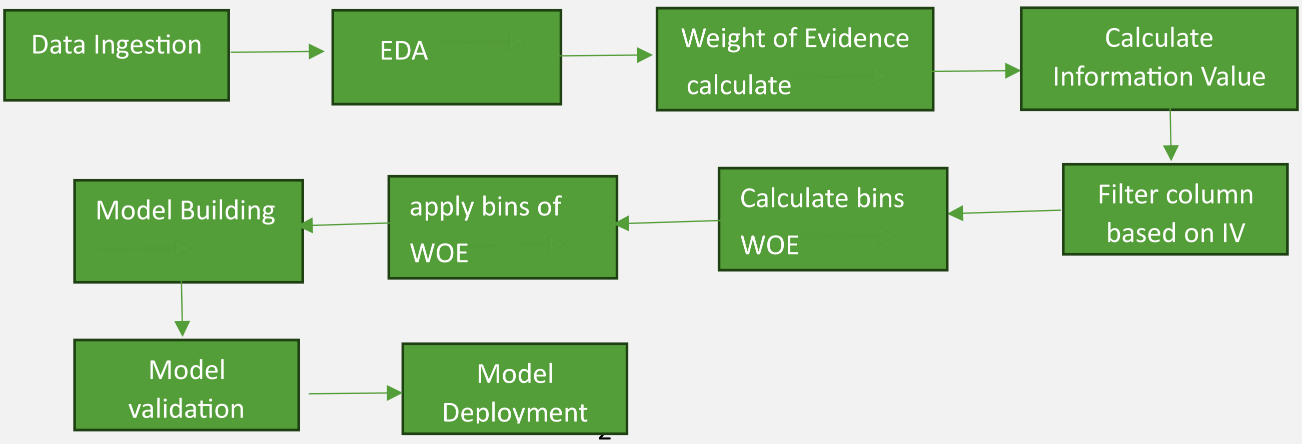
The problem of credit card default is a significant concern for both financial institutions and borrowers. Default occurs when a credit cardholder fails to make the required minimum payments on their credit card balance within a specified timeframe. This situation poses significant financial risks to credit card companies and can lead to substantial losses. Therefore, accurately predicting the likelihood of credit card default is crucial for mitigating these risks and making informed decisions regarding credit card approvals, credit limits, and interest rates.

# Proposed Solution:

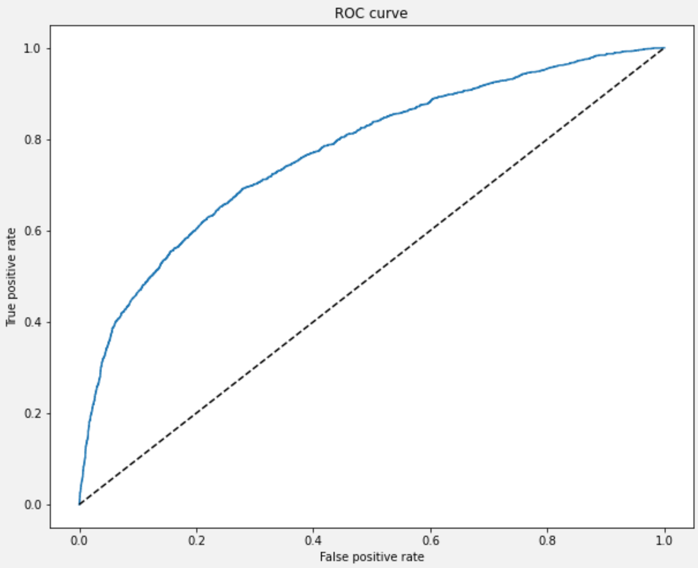
The project follows a supervised learning approach, utilizing a dataset containing a comprehensive range of features, including credit limit, payment history, amount owed, age, etc. Various machine learning algorithms, such as logistic regression, decision trees, random forests, and gradient boosting, were explored and compared. Feature engineering techniques, including Weight of Evidence, were employed to enhance model performance. AUC ROC GINI are the metrics used for model validation

# Technical Requirements and Tools:

# Architecture Pipeline :



# Model Result and Model tuning




```python
# Calculate the Area Under the Receiver Operating Characteristic Curve (AUROC) on our test set
auroc = roc_auc_score(y_test_proba['y_test_class_actual'], y_test_proba['y_hat_test_proba'])
auroc
```

```
0.7712918571428573
```

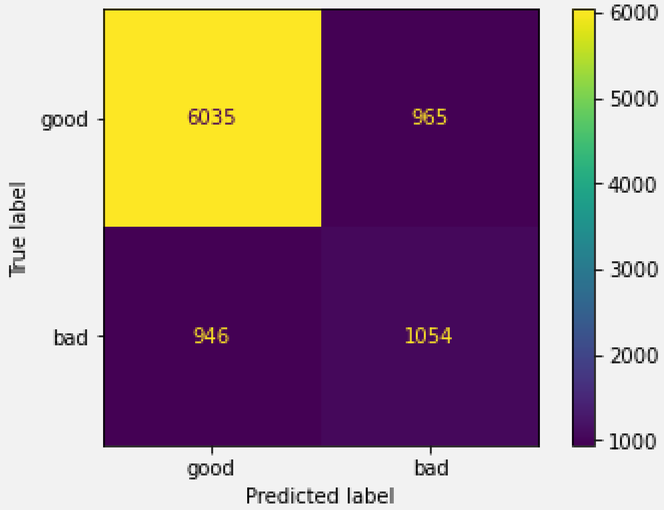
```python
# calculate Gini from AUROC
gini = auroc * 2 - 1
gini
```

```
0.5425837142857146
```

```
Optimization terminated successfully.
         Current function value: 0.556409
         Iterations 6
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: -0.054
Dependent Variable: target           AIC:              23399.1647
Date:               2023-05-24 02:43 BIC:              23518.4489
No. Observations:   21000            Log-Likelihood:   -11685.
Df Model:           14               LL-Null:          -11085.
Df Residuals:       20985            LLR p-value:      1.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
-----------------------------------------------------------------
                   Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
-----------------------------------------------------------------
BILL_AMT3_woe     -0.4137   0.2781 -1.4874 0.1369 -0.9589  0.1314
AGE_woe            0.4506   0.1125  4.0067 0.0001  0.2302  0.6710
PAY_2_woe          0.0933   0.0392  2.3801 0.0173  0.0165  0.1702
PAY_AMT4_woe       0.1298   0.0676  1.9191 0.0550 -0.0028  0.2623
PAY_3_woe          0.2548   0.0424  6.0129 0.0000  0.1717  0.3378
PAY_0_woe          0.9460   0.0286 33.0886 0.0000  0.8900  1.0020
BILL_AMT1_woe     -1.1853   0.2717 -4.3631 0.0000 -1.7177 -0.6528
PAY_AMT3_woe       0.1545   0.0609  2.5360 0.0112  0.0351  0.2738
PAY_6_woe          0.3311   0.0473  7.0009 0.0000  0.2384  0.4239
...
PAY_AMT6_woe       0.1837   0.0648  2.8338 0.0046  0.0566  0.3107
PAY_AMT2_woe       0.1808   0.0528  3.4263 0.0006  0.0774  0.2842
=================================================================
```

After dropping the variables, the achieved result has been improved. Accuracy , precision, recall auc and gini index has been checked with respect to different threshold. I have applied probability 0.10 to 0.50 and 0.26 came out the optimal for the modelling with accuracy 0.787889, precision 0.522403, recall 0.5305, ruc 0.695964 and gini 0.391929. This is clearly seen that recall value has improved from 0.35 to 0.53 and AUC from 0.64 to 0.69

| | threshold | accuracy | precision | recall | auroc | gini |
|---|---|---|---|---|---|---|
| 0 | 0.10 | 0.415667 | 0.267248 | 0.9355 | 0.601321 | 0.202643 |
| 1 | 0.15 | 0.654222 | 0.365245 | 0.7535 | 0.689679 | 0.379357 |
| 2 | 0.20 | 0.755556 | 0.461744 | 0.6035 | 0.701250 | 0.402500 |
| 3 | 0.25 | 0.783667 | 0.512625 | 0.5380 | 0.695929 | 0.391857 |
| 4 | 0.26 | 0.787667 | 0.522041 | 0.5270 | 0.694571 | 0.389143 |
| 5 | 0.27 | 0.789444 | 0.526827 | 0.5155 | 0.691607 | 0.383214 |
| 6 | 0.28 | 0.791889 | 0.533229 | 0.5095 | 0.691036 | 0.382071 |
| 7 | 0.29 | 0.794444 | 0.540279 | 0.5030 | 0.690357 | 0.380714 |
| 8 | 0.30 | 0.796111 | 0.545656 | 0.4930 | 0.687857 | 0.375714 |
| 9 | 0.31 | 0.798778 | 0.554031 | 0.4845 | 0.686536 | 0.373071 |
| 10 | 0.32 | 0.800111 | 0.559292 | 0.4740 | 0.683643 | 0.367286 |
| 11 | 0.33 | 0.801778 | 0.565534 | 0.4660 | 0.681857 | 0.363714 |
| 12 | 0.34 | 0.807222 | 0.585209 | 0.4550 | 0.681429 | 0.362857 |
| 13 | 0.35 | 0.809000 | 0.594486 | 0.4420 | 0.677929 | 0.355857 |
| 14 | 0.40 | 0.818000 | 0.643879 | 0.4050 | 0.670500 | 0.341000 |
| 15 | 0.45 | 0.818333 | 0.664563 | 0.3685 | 0.657679 | 0.315357 |
| 16 | 0.50 | 0.816667 | 0.675351 | 0.3370 | 0.645357 | 0.290714 |

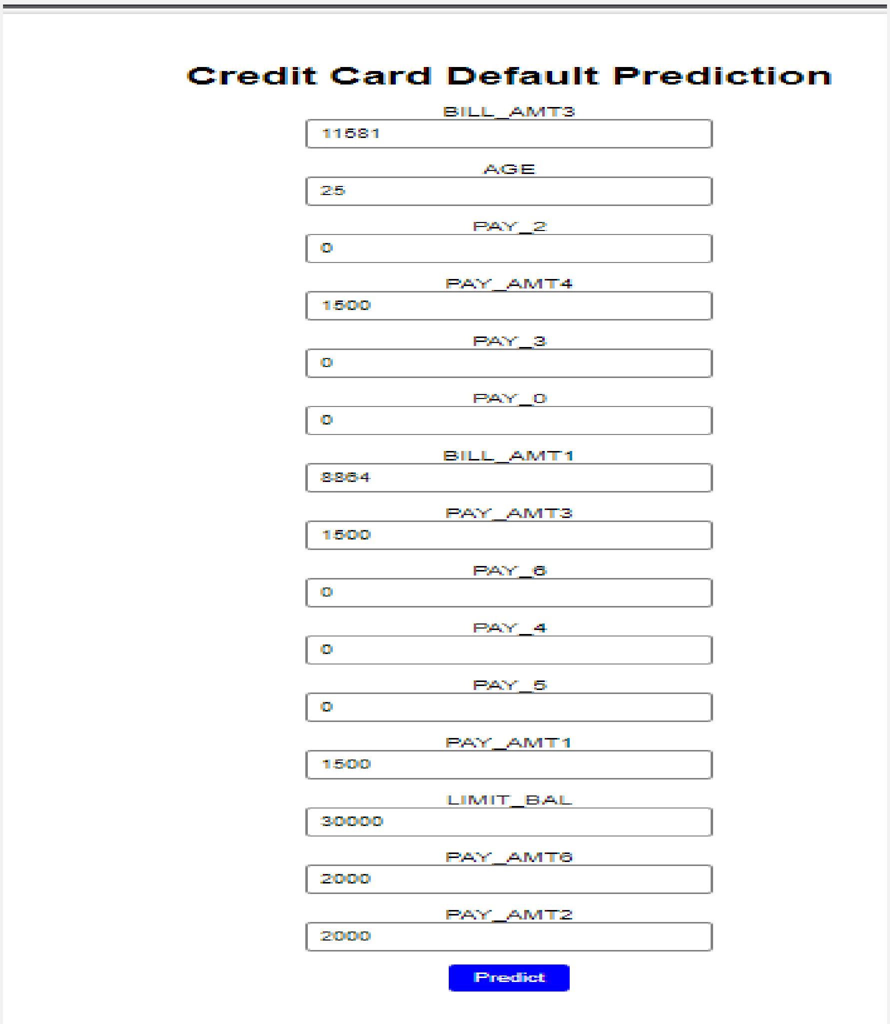


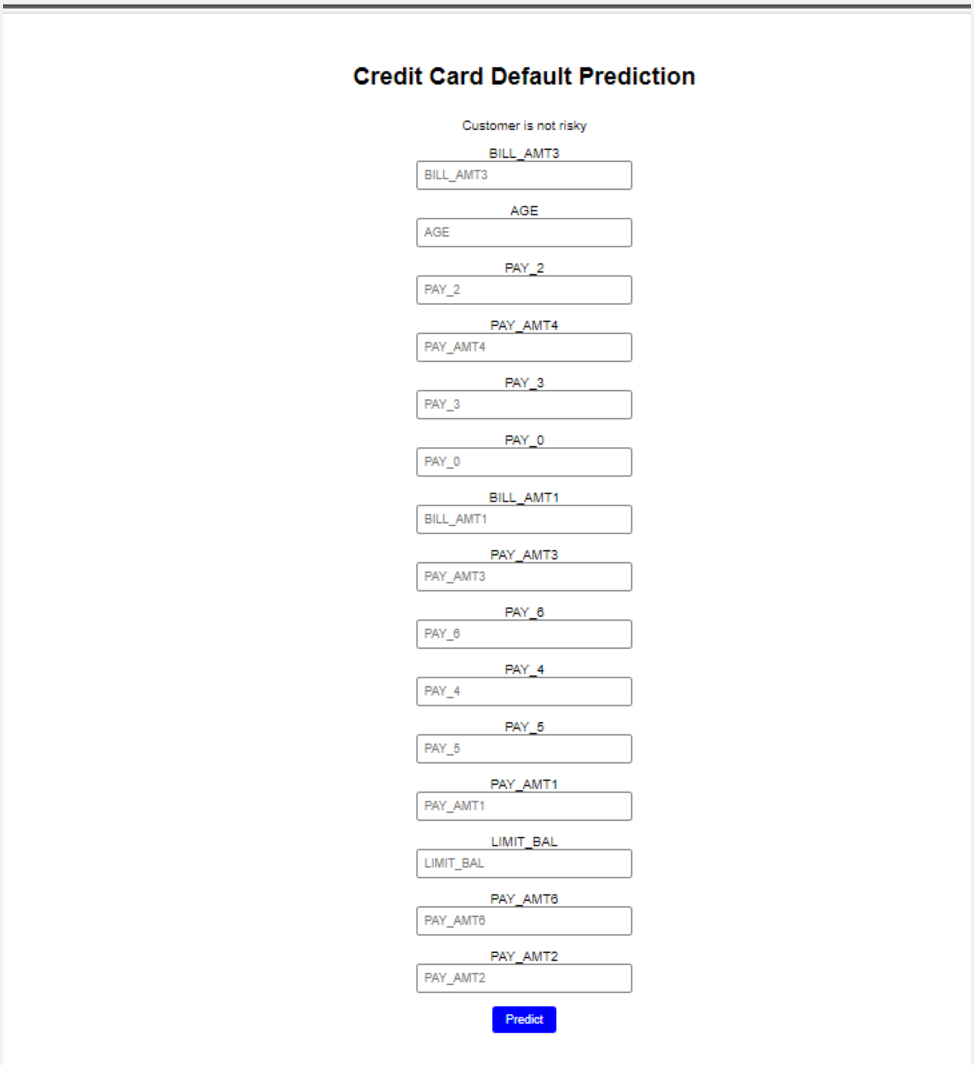

Final cutoff has been chosen on 0.26 threshold. So if model throw probability more than 0.26 then model will decide the customer is a risky customer other wise it declare non risky customer

# Model Deployment:

This model has deployed into AWS using Flask with CICD pipeline process.

1. Create Docker Image
2. Create an instance for deploying
3. Create an IAM user
4. Create ECR repository
5. Set GitHub actions
6. Set AWS credentials in GitHub




Credit Card Default Prediction
Customer is not risky
BILL_AMT3
BILL_AMT3
AGE
AGE
PAY_2
PAY_2
PAY_AMT4
PAY_AMT4
PAY_3
PAY_3
PAY_0
PAY_0
BILL_AMT1
BILL_AMT1
PAY_AMT3
PAY_AMT3
PAY_6
PAY_6
PAY_4
PAY_4
PAY_5
PAY_5
PAY_AMT1
PAY_AMT1
LIMIT_BAL
LIMIT_BAL
PAY_AMT6
PAY_AMT6
PAY_AMT2
PAY_AMT2
Predict