Bank Credit Risk Classification

Architecture

Shruti Balan
18 October, 2022

Contents

1. Introduction

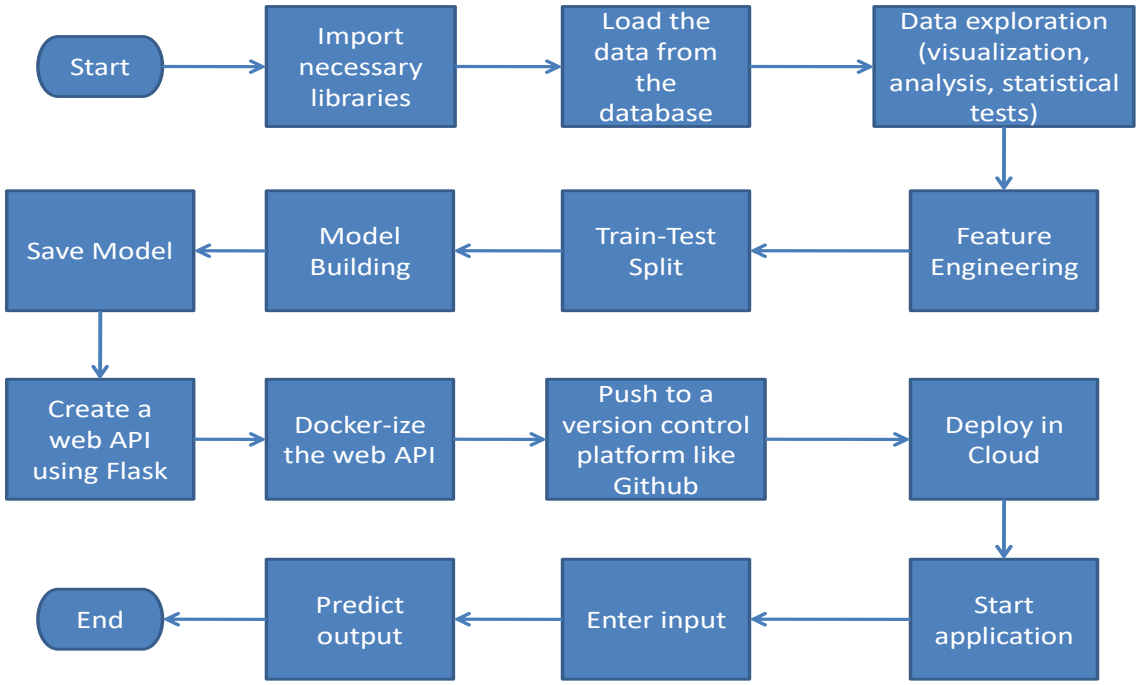
1.1. What is Architectural design document?

The purpose of this document is to provide a comprehensive architectural overview of the Bank Credit Risk Classification application. It will explain the architecture and features, the modules and components of the application. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher management for its approval.

1.2. Scope

The architectural design is a map of the software. It allows the users to see the structure of the software at a glance.

2. Architecture



3. Architecture Description

3.1. Data Description

This dataset is taken from the UCI Machine Learning Repository (url: https://archive.ics.uci.edu/ml/datasets/South+German+Credit). It contains information on defaults, demographic factors, credit data etc. of customers. There are 21 variables:

Content There are 21 variables:

- **status** : status of the debtor's checking account with the bank (categorical)
  - **1** : no checking account
  - **2** : ... < 0 DM
  - **3** : 0<= ... < 200 DM
  - **4** : ... >= 200 DM / salary for at least 1 year

- **duration** : credit duration in months (quantitative)

- **credit_history** : history of compliance with previous or concurrent credit contracts (categorical)
  - **0** : delay in paying off in the past
  - **1** : critical account/other credits elsewhere
  - **2** : no credits taken/all credits paid back duly
  - **3** : existing credits paid back duly till now
  - **4** : all credits at this bank paid back duly

- **purpose** : purpose for which the credit is needed (categorical)
  - **0** : others
  - **1** : car (new)
  - **2** : car (used)
  - **3** : furniture/equipment
  - **4** : radio/television
  - **5** : domestic appliances
  - **6** : repairs
  - **7** : education
  - **8** : vacation
  - **9** : retraining
  - **10** : business
- **amount** : credit amount in DM (quantitative; result of monotonic transformation; actual data and type of transformation unknown)

- **savings** : debtor's savings (categorical)
  - **1** : unknown/no savings account
  - **2** : ... < 100 DM
  - **3** : 100 <= ... < 500 DM
  - **4** : 500 <= ... < 1000 DM
  - **5** : ... >= 1000 DM

- **employment_duration** : duration of debtor's employment with current employer (ordinal; discretized quantitative)
    - **1** : unemployed
    - **2** : < 1 yr
    - **3** : 1 <= ... < 4 yrs
    - **4** : 4 <= ... < 7 yrs
    - **5** : >= 7 yrs

- **installment_rate** : credit installments as a percentage of debtor's disposable income (ordinal; discretized quantitative)
    - **1** : >= 35
    - **2** : 25 <= ... < 35
    - **3** : 20 <= ... < 25
    - **4** : < 20

- **personal_status_sex** : combined information on sex and marital status; categorical; sex cannot be recovered from the variable, because male singles and female non-singles are coded with the same code (2); female widows cannot be easily classified, because the code table does not list them in any of the female categories
    - **1** : male : divorced/separated
    - **2** : female : non-single or male : single
    - **3** : male : married/widowed
    - **4** : female : single

- **other_debtors :** Is there another debtor or a guarantor for the credit? (categorical)
    - **1** : none
    - **2** : co-applicant
    - **3** : guarantor

- **present_residence :** length of time (in years) the debtor lives in the present residence (ordinal; discretized quantitative)
    - **1** : < 1 yr
    - **2** : 1 <= ... < 4 yrs
    - **3** : 4 <= ... < 7 yrs
    - **4** : >= 7 yrs

- **property :** the debtor's most valuable property, i.e. the highest possible code is used. Code 2 is used, if codes 3 or 4 are not applicable and there is a car or any other relevant property that does not fall under variable savings. (ordinal)
    - **1** : unknown / no property
    - **2** : car or other
    - **3** : building soc. savings agr./life insurance
    - **4** : real estate
- **age :**age in years (quantitative)

- **other_installment_plans :**installment plans from providers other than the credit-giving bank (categorical)
  - **1** : bank
  - **2** : stores
  - **3** : none

- **housing :**type of housing the debtor lives in (categorical)
  - **1** : for free
  - **2** : rent
  - **3** : own

- **number_credits :**number of credits including the current one the debtor has (or had) at this bank (ordinal, discretized quantitative); contrary to Fahrmeir and HamerleÃ¢â‚¬â„¢s (1984) statement, the original data values are not available.
  - **1** : 1
  - **2** : 2-3
  - **3** : 4-5
  - **4** : >= 6

- **job :**quality of debtor's job (ordinal)
  - **1** : unemployed/unskilled - non-resident
  - **2** : unskilled - resident
  - **3** : skilled employee/official
  - **4** : manager/self-empl./highly qualif. employee

- **people_liable :**number of persons who financially depend on the debtor (i.e., are entitled to maintenance) (binary, discretized quantitative)
  - **1** : 3 or more
  - **2** : 0 to 2

- **telephone :**Is there a telephone landline registered on the debtor's name? (binary; remember that the data are from the 1970s)
  - **1 :** No
  - **2 :** Yes

- **foreign_worker :**Is the debtor a foreign worker? (binary)
  - **1** : yes
  - **2** : no

- **credit_risk :**Has the credit contract been complied with (good) or not (bad) ? (binary)
  - **0** : bad
  - **1** : good

3.2. Data Exploration

We divide the data into two types: numerical and categorical. We explore through each type one by one. Within each type, we explore, visualize and analyze each variable one by one

and note down our observations. Statistical tests are performed for each independent variable to see if the variable has any significance in determining the output for the target variable.

3.3. Feature Engineering
Encoded categorical variables.

3.4. Train/Test Split
Split the data into 70% train set and 30% test set.

3.5. Model Building
Built models and trained and tested the data on the models.
Compared the performance of each model and selected the best one.
Feature importance and/or hyper-parameter tuning performed to improve the performance of the selected model.

3.6. Save the model
Saved the model by converting into a pickle file.

3.7. Create a web API
The model is used to create a webAPI using which the users can interact with the application. In this project, Flask has been used for the purpose.

3.8. Docker-ize the application
The application is then containerized using Docker.

3.9. Cloud Setup & Pushing the App to the Cloud
Selected Heroku for deployment.
Used the model to develop a flask application which can predict risk class for unseen data.
Dockerised the application and pushed to Github using and from there deployed the application files from Github to Heroku, all using Github Actions as ci/cd pipeline.

3.10. Application Start and Input Data by the User
Start the application and enter the inputs.

3.11. Prediction
After the inputs are submitted the application runs the model and makes predictions. The out is displayed as a message indicating whether the customer is classified as Good Risk or Bad Risk.

4. Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether user is able to see input fields on logging in	1. Application URL is accessible 2. Application is deployed	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application URL is accessible 2. Application is deployed	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application URL is accessible 2. Application is deployed	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application URL is accessible 2. Application is deployed	User should be presented with recommended results on clicking submit