Architecture Design

Credit Risk Predictor

Revision Number: 1.0
Last date of revision: 1/8/23

Pothuri Harish Varma

Document Version Control

Date	Version	Description	Author
01-08-2023	V1.0	Architecture Design-V1.0	Pothuri Harish Varma

Contents

1. Introduction

1.1. Why this Architecture Design Document?

The goal of Architecture Design (AD) is to give the internal design of the actual program code for the 'Prediction Bank Credit Risk Using South German Credit Data '. AD describes the class diagrams with the methods and relation between classes and program specification. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Architecture Design (AD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software, architecture, source code, and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work. And the complete workflow.

1.3. Constraints

We predict the expected estimating cost of expenses customers based on some personal health information.

2. Technical Specification

2.1. Dataset

The dataset contains verified historical data, consisting of the aforementioned information which classifies people described by a set of attributes as good or bad credit risks. There are 1000 entries in the dataset, which have been sampled as train and test data. This is an adaptation of the UCI South German Credit Dataset. The objective is to find a way to classify the good and bad credit risk identified as 1 and 0 respectively using values in the other feature columns like their status, duration, credit history, purpose, amount, savings, employment duration, installment rate, personal status sex, other debtors, present residence, property, age, other installment plans, housing, number credits, job, people liable, telephone, foreign worker, credit risk.

The dataset looks like as follows:

Import the ASC Data as Pandas DataFrame

In [2]:
```
import pandas as pd
df = pd.read_csv("C:/project1/notebook/data/SouthGermanCredit.asc", sep=' ')
df
```

Out[2]:

	laufkont	laufzeit	moral	verw	hoehe	sparkont	beszeit	rate	famges	buerge	...	verm	alter	weitkred	wohn	bishkred	beruf	pers	telef	gastarb
0	1	18	4	2	1049	1	2	4	2	1	...	2	21	3	1	1	3	2	1	2
1	1	9	4	0	2799	1	3	2	3	1	...	1	36	3	1	2	3	1	1	2
2	2	12	2	9	841	2	4	2	2	1	...	1	23	3	1	1	2	2	1	2
3	1	12	4	0	2122	1	3	3	3	1	...	1	39	3	1	2	2	1	1	1
4	1	12	4	0	2171	1	3	4	3	1	...	2	38	1	2	2	2	2	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	1	24	2	3	1987	1	3	2	3	1	...	1	21	3	1	1	2	1	1	2
996	1	24	2	0	2303	1	5	4	3	2	...	1	45	3	2	1	3	2	1	2
997	4	21	4	0	12680	5	5	4	3	1	...	4	30	3	3	1	4	2	2	2

The data set consists of various data types from integer to floating to object as shown in Figure:

```
# Check Null and Dtypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   status                   1000 non-null   int64
 1   duration                 1000 non-null   int64
 2   credit_history           1000 non-null   int64
 3   purpose                  1000 non-null   int64
 4   amount                   1000 non-null   int64
 5   savings                  1000 non-null   int64
 6   employment_duration      1000 non-null   int64
 7   installment_rate         1000 non-null   int64
 8   personal_status_sex      1000 non-null   int64
 9   other_debtors            1000 non-null   int64
 10  present_residence        1000 non-null   int64
 11  property                 1000 non-null   int64
 12  age                      1000 non-null   int64
 13  other_installment_plans  1000 non-null   int64
 14  housing                  1000 non-null   int64
 15  number_credits           1000 non-null   int64
 16  job                      1000 non-null   int64
 17  other_installment_plans  1000 non-null   int64
 18  telephone                1000 non-null   int64
 19  foreign_worker           1000 non-null   int64
 20  credit_risk              1000 non-null   int64
dtypes: int64(21)
memory usage: 164.2 KB
```

Various factors important by statistical means like mean, standard deviation, median, count of values and maximum value, etc. are shown below for numerical attributes

3.5 Check statistics of data set

```
]: df.describe()
```

t[35]:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	other_
count	1000.000000	1000.000000	1000.00000	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000	1000.00000	1000
mean	2.577000	20.903000	2.54500	2.828000	3271.24800	2.105000	3.384000	2.973000	2.68200	1
std	1.257638	12.058814	1.08312	2.744439	2822.75176	1.580023	1.208306	1.118715	0.70808	0
min	1.000000	4.000000	0.00000	0.000000	250.00000	1.000000	1.000000	1.000000	1.00000	1
25%	1.000000	12.000000	2.00000	1.000000	1365.50000	1.000000	3.000000	2.000000	2.00000	1
50%	2.000000	18.000000	2.00000	2.000000	2319.50000	1.000000	3.000000	3.000000	3.00000	1
75%	4.000000	24.000000	4.00000	3.000000	3972.25000	3.000000	5.000000	4.000000	3.00000	1
max	4.000000	72.000000	4.00000	10.000000	18424.00000	5.000000	5.000000	4.000000	4.00000	3

8 rows × 21 columns

Pre-processing of this dataset includes doing analysis on the independent variables like checking for null values in each column and then replacing them with supported appropriate data types so that analysis and model fitting is not hindered from their way to accuracy. Shown above are some of the representations obtained by using Pandas tools which tells about variable count for numerical columns and model values for categorical columns. Maximum and minimum values in numerical columns, along with their percentile values for median, plays an important factor in deciding which value to be chosen at priority for further exploration tasks and analysis. Data types of different columns are used further in label processing and a one-hot encoding scheme during the model building.

2.2. Logging

We should be able to log every activity done by the user

- The system identifies at which step logging is required.
- The system should be able to log each and every system flow.
- The system should not be hung even after using so much logging. Logging just because we can easily debug issuing so logging is mandatory to do.

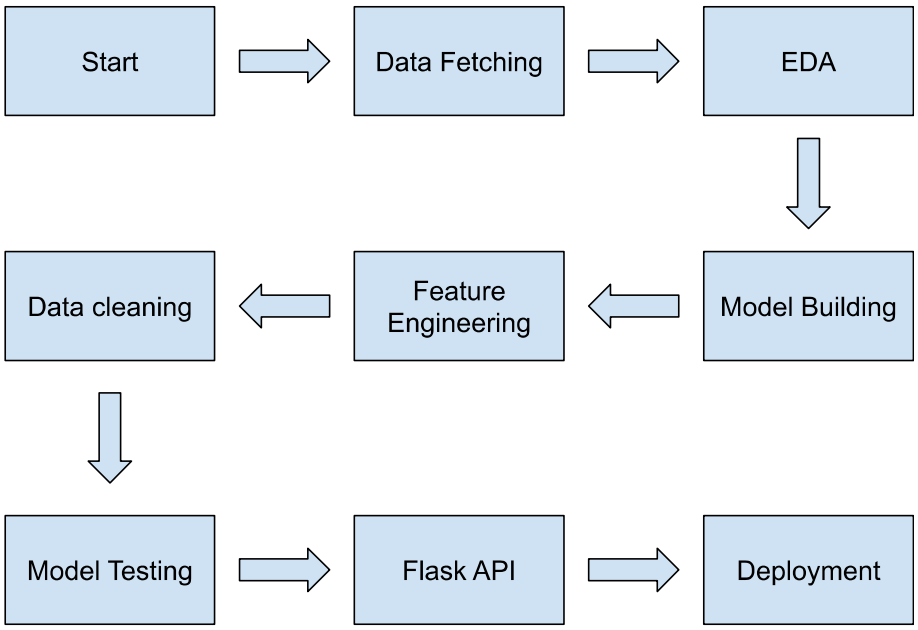
3. Technology Stack

Front End	Html
Back End	Python/Flask
Deployment	Amazon Web Services

4. Proposed Solution

The solution of this problem statement is to perform EDA on the dataset to generate meaningful insights from the data and use this data to hyper tune with appropriate machine learning algorithms which will have the maximum accuracy in predicting the credit risk. Thus creating a user interface where a user can put in the various features of the data which will in return give the credit risk is present or not.

5. Architecture



5.1. Data Gathering

Data source: https://archive.ics.uci.edu/datasets?search=credit+risk Dataset is stored in ASC format

5.2. Raw Data Validation

After data is loaded, various types of validation are required before we proceed further with any operation. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because the attributes which contain these are of no use. It will not play role in contributing to the estimating the Good and Bad credit risk i.e. 1 and 0 respectively.

5.3. Exploratory Data Analysis

Visualized the relationship between the dependent and independent features. Also checked relationship between independent features to get more insights about the data.

5.4. Model Building

After doing all kinds of preprocessing operations mention above and performing scaling and encoding, the data set is passed through a pipeline to all the mode Bagging, Random Forest, Gradient boost, Xgboost, Adaboost Classifier algorithms

5.5. Model Saving

Model is saved using the pickle library in pickle format.

5.6. Flask API for Web Application

After saving the model, the API building process started using Flask . Web application creation was created in Flask for testing purposes. Whatever user will enter the data and then that data will be extracted by the model to estimate the bank credit risk good or bad i.e. 1 or 0 respectively, this is performed in this stage.

5.7. GitHub

The whole project directory will be pushed into the GitHub repository.

5.8. Deployment

The Model was Deployed in AWS ECR Instance.

6. User Input / Output Workflow