Architecture

Credit Risk System

Revision Number: 1.0
Last date of revision: 22/12/2021

Jeevan Shriram Arande

Raj Shukla

1. Introduction

Normally, most of the bank's wealth is obtained from providing credit loans so that a marketing bank must be able to reduce the risk of non-performing credit loans. The risk of providing loans can be minimized by studying patterns from existing lending data. One technique that you can use to solve this problem is to use data mining techniques. Data mining makes it possible to find hidden information from large data sets by way of classification.

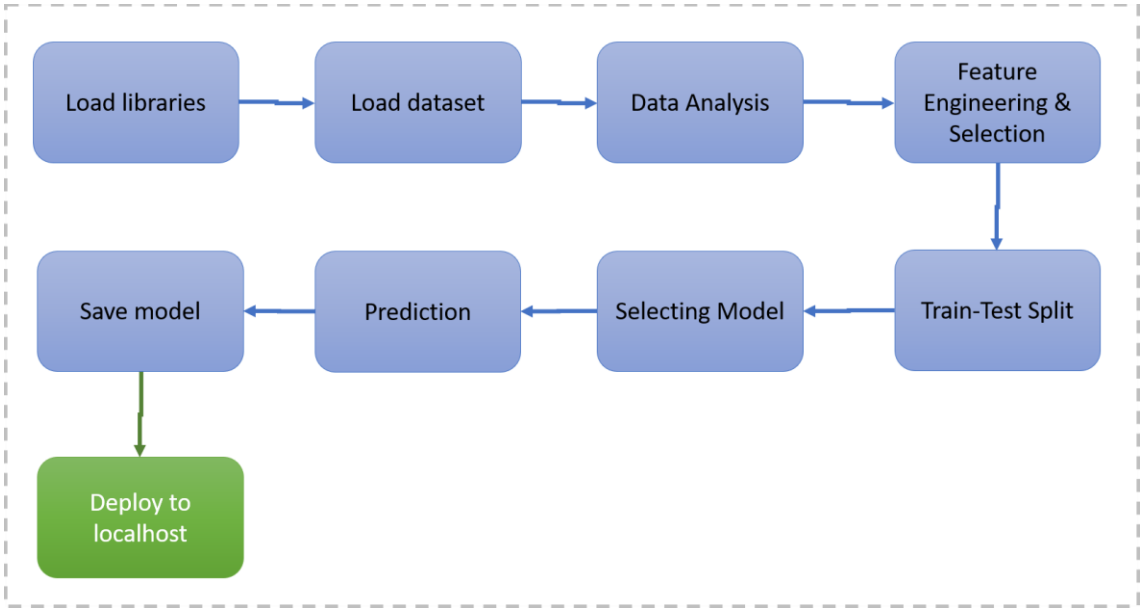
The goal of this project, you have to build a model to predict whether the person, described by the attributes of the dataset, is a good (1) or a bad (0) credit risk

2. Problem Statement

To create an ML solution for Credit Risk Prediction and to implement the following use case:

1. Based on the existing current customer risk data, build a model to predict a credit risk profile (Good or Bad).
2. Mitigate the risk by not providing loans to people with non-performing loans.
3. Provide loans to customers with good credit history.

3. Training Architecture



3.1. Data Description

The widely used Statlog German credit data ([Web Link]), as of November 2019, suffers from severe errors in the coding information and does not come with any background information. The 'South German Credit' data provide a correction and some background information, based on the Open Data LMU (2010) representation of the same data and several other German language resources.

3.2. Data Pre-processing

This included importing of important libraries such as matplotlib, pandas, sklearn etc. We imported the same dataset mentioned above.

3.3. Data Analysis

Here we handled the null values, changed the column names, plotted multiple graphs in matplotlib and other visualization library for proper understanding of the data and the distribution of information in the same. As there were no null values in the data, we proceeded with the visualization and analysis.

3.4. Feature Engineering & Selection

Feature Selection using ANOVA for continuous variables and Chi Square Analysis for Categorical columns. We converted the nominal categorical columns using one-hot encoding and scaled the data using Min Max Scalar.

3.5. Train/Test Split

This library was imported from Sklearn to divide the final dataset into the ratio of 70-30%, where 70% of the data was used to train the model and the latter 30% was used to predict the same.

3.6. Selecting Model

We tried and tested multiple models such as XGBoost, RandomForest, Decision Tree, Logistic Regression and Naïve Bayes for the model and came up with the model with the best performance, i.e the Random Forest Classifier.

3.7. Prediction

The Accuracy of Random Forest was 61% and the F1 score was 69%.

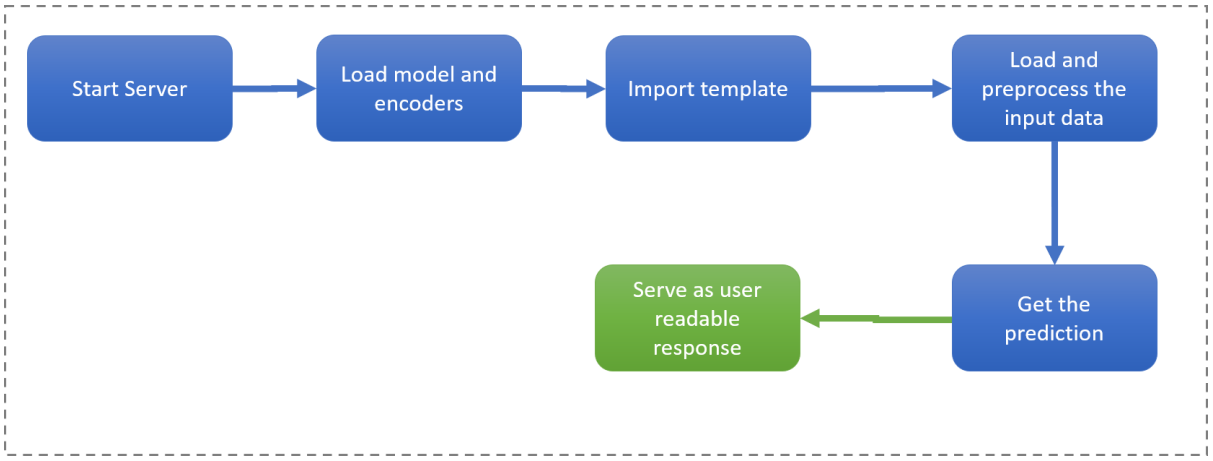
3.8 Save Model

Model and encoders were saved using the pickle library which saves the file in a binary mode.

3.9 Deploy in Local Host

We created a HTML template and deployed the model through Flask.

4. Deployment Architecture



4.1. Start Server

Start the Flask server for accepting API calls using Post Method and and clean the data and get the JSON dictionary.

4.2. Model and Encoder loading

Load the fitted model and encoders from binary pickle files

4.3. Import template

Import HTML template containing the form to get the data from the user as a POST request

4.4. Load and pre-process the data from API request

Load and clean the data from the post request, convert it into dictionary and following the pre-processing steps

4.5 Get the prediction

Use the loaded model to get the prediction (0 – Bad & 1 - Good)

4.6 Serve as user readable response

Pass the response as “credit_risk”: “good” or “credit_risk”: “bad” (As JSON Dict)