Loan Eligibility prediction
Project overview

Business Objective

When a customer applies for a loan at our company, we use statistical models to determine whether or not to grant the loan based on the likelihood of the loan being repaid. The factors involved in determining this likelihood are complex, and extensive statistical analysis and modelling are required to predict the outcome for each individual case.

Aim

You must implement a model that predicts if a loan should be granted to an individual based on the data provided

Tech Stack

- ➔ Language : Python
- ➔ Libraries : Scikit-learn, H2O, pandas, numpy, flask, Seaborn, Matplotlib
- ➔ Containerization : Docker

Dataset Description

The dataset used is an anonymized synthetic data that was generated specifically for use in this project. The data is designed to exhibit similar characteristics to genuine loan data.

In this dataset, you must explore and cleanse a dataset consisting of over 1,00,000 loan records to determine the best way to predict whether a loan applicant should be granted a loan or not.

Data files

The dataset consists of the following fields:

1. Loan ID: A unique Identifier for the loan information.
2. Customer ID: A unique identifier for the customer. Customers may have more than one loan.
3. Loan Status: A categorical variable indicating if the loan was given to this customer
4. Current Loan Amount: This is the loan amount that was either completely paid off, or the amount that was defaulted. This data is for previous loan
5. Term: A categorical variable indicating if it is a short term or long term loan.
6. Credit Score: A value between 0 and 800 indicating the riskiness of the borrower's credit history.
7. Years in current job: A categorical variable indicating how many years the customer has been in their current job.

8. Home Ownership: Categorical variable indicating home ownership. Values are "Rent", "Home Mortgage", and "Own". If the value is Own, then the customer is a homeowner with no mortgage
9. Annual Income: The customer's annual income
10. Purpose: A description of the purpose of the loan.
11. Monthly Debt: The customer's monthly payment for their existing loansYears of Credit History: The years since the first entry in the customer's credit history •
12. Months since last delinquent: Months since the last loan delinquent payment
13. Number of Open Accounts: The total number of open credit cards
14. Number of Credit Problems: The number of credit problems in the customer records.
15. Current Credit Balance: The current total debt for the customer
16. Maximum Open Credit: The maximum credit limit for all credit sources.
17. Bankruptcies: The number of bankruptcies
18. Tax Liens: The number of tax liens.

Approach

1. EDA
    - Missing data analysis
    - Data imputation
    - Bad features removal
    - Features boxplot and distribution analysis basis visualization

2. Data cleaning / Pre-processing (outlier/missing values/categorical)
    - One Hot encoder for categorical variables
    - Label encoder if feature needs to be discretized

3. Feature Engineering
    - Non-linear combination of existing features
    - Add pre-calculated priors as features

4. Standardizing data
    - Standard scaler

5. Modelling
    - Random Forest
    - Boosting Models
    - Light GBM
    - XGBoost

- GBT
- Neural Network & variants

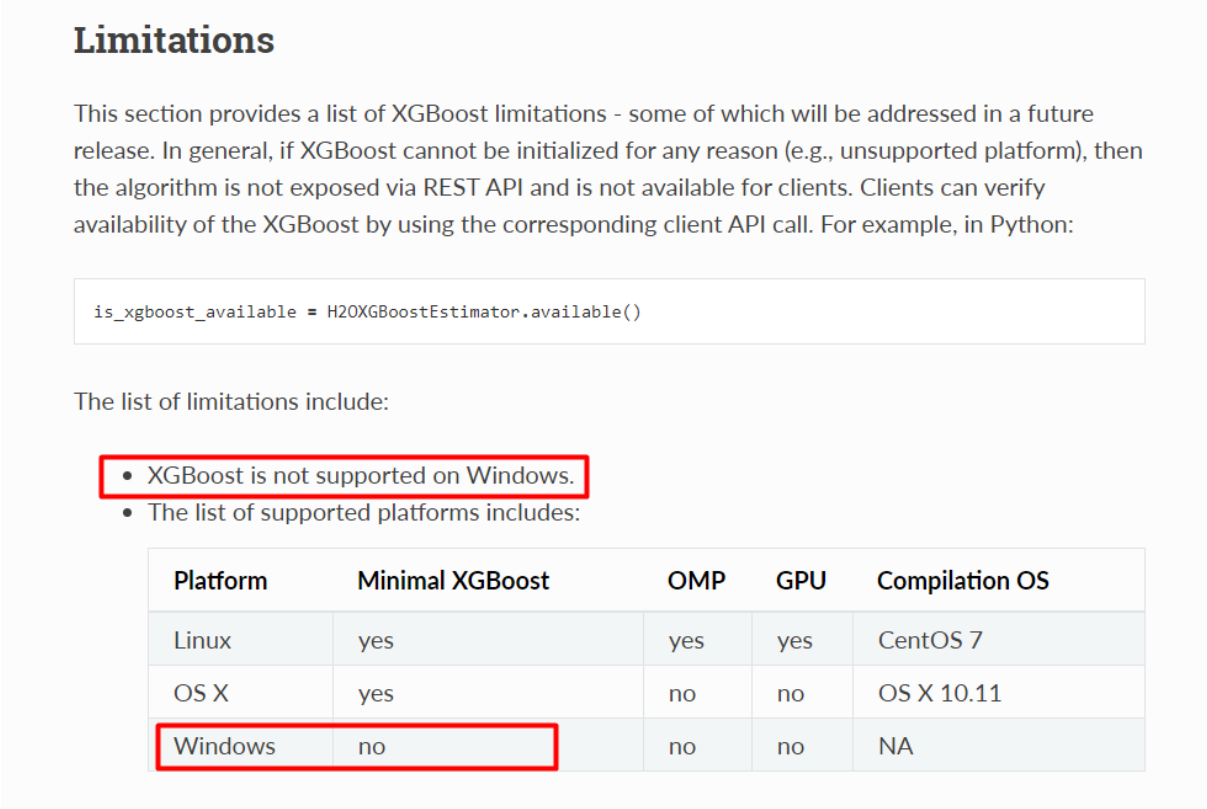
6. Hyper parameter tuning
    - GridSearchCV to find out the best model

7. Finalise Model - Model which has the best AUCPR on the labeled test data would be used. Although AUC and F1 score would also be tracked, the following metrics are also taken into account
    - Precision/Recall
    - AUC
    - AUCPR
    - F1 Score

8. Deployment
    - Deployment using a Flask API and with a Docker container

Note

If you are using Windows OS, the XGBoost model might not work in your system since H2O-3.32.1.2 does not support XGBoost for windows. Please refer to this link for further details.



Limitations

This section provides a list of XGBoost limitations - some of which will be addressed in a future release. In general, if XGBoost cannot be initialized for any reason (e.g., unsupported platform), then the algorithm is not exposed via REST API and is not available for clients. Clients can verify availability of the XGBoost by using the corresponding client API call. For example, in Python:

```
is_xgboost_available = H2OXGBoostEstimator.available()
```

The list of limitations include:

- XGBoost is not supported on Windows.
- The list of supported platforms includes:

Platform	Minimal XGBoost	OMP	GPU	Compilation OS
Linux	yes	yes	yes	CentOS 7
OS X	yes	no	no	OS X 10.11
Windows	no	no	no	NA

Modular code overview

```
input
  |__final_data.csv
  |__grouped_by_loan_id.csv
  |__LoansTrainingSetV2.csv
  |__test_data.csv
src
  |__engine.py
  |__ML_pipeline
            |__categorical_encoding.py
            |__data_scaling.py
            |__deep_learning_h2o.py
            |__gb_model_h2o.py
            |__light_gbm_h2o.py
            |__xgb_model_h2o.py
            |__train_test_h2o.py
            |__utils.py
            |__impute.py
            |__knn_imputation.py
  |__deployment
            |__Deploy.py
            |__Dockerfile
            |__Model_Api.ipynb
            |__server.py
lib
  |__Multiple-Classification-h2o.ipynb
output
  |__DeepLearning_model_python_1621429680734_1065
  |__GBM_model_python_1621429680734_905
  |__gradient_boosting_model.hex
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input
2. src
3. output
4. lib

1. The input folder contains all the data that we have for analysis.
2. The src folder is the heart of the project. This folder contains all the modularized code for all the above steps in a modularized manner. It further contains the following.
    a. ML_pipeline
    b. deployment
    c. engine.py

The ML_pipeline is a folder that contains all the functions put into different python files which are appropriately named. These python functions are then called inside the engine.py file. The deployment folder contains all the files related to deploying the model

3. The output folder contains all the models that we trained for this data saved as reusable files. These models can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
4. The lib folder is a reference folder. It contains the original ipython notebook that we saw in the videos.

Project Takeaways

1. Data Analysis using EDA
2. Features boxplot and distribution analysis basis visualization
3. Label encoding and Onehot encoding
4. Adding derived features from existing features
5. KNN imputation
6. Standard Scaling
7. Building Gradient boosting model using H2o
8. Building XGBoost model using H2o
9. Building Light GBM using H2O
10. Building Deep learning models in H2O
11. Understanding model metrics such as Precision/Recall, AUC, AUCPR, F1 Score
12. Drawing predictions from the models
13. Dockerizing ML models
14. Deploying ML models using flask