Auxiliary code documentation

Interest Rate Not Guaranteed team

March 2024

1 Libraries used

We used the following libraries in the project:

- **numpy** - version 1.26.0
- **pandas** - version 2.1.4
- **seaborn** - version 0.13.2
- **sklearn** - version 1.3.2
- **statsmodels** - version 0.14.1
- **scipy** - version 1.11.4

2 Data transformation

The whole process of transforming the data for further analysys is handled through the `transform` function which is defined in the `data_transfomation.py` file. The function's key features are:

Column Renaming:
Renames the columns of the dataset based on the provided mapping.

Handling Missing Values:
Fills missing values in certain columns with appropriate strategies (median for numerical columns, specific values for categorical columns).

Feature Engineering:
Calculates new features like `yearly_spend_ratio, zero_savings, zero_income, loan_to_income, secondary_applicant, empl_to_app_time`, etc.

Data Type Conversion:
Converts certain columns to `datetime` format using custom conversion functions. Bins `credit_score` into buckets.

One-Hot Encoding: Performs one-hot encoding on selected categorical columns.

Output: Returns the transformed `DataFrame`.

Each of the following codes will work on data transformed by the aforementioned `transform` function.

3 Model development

The development of the logistic regression model is defined in the `model_tuning.ipynb` file.

After the imports a constant column is added to the feature matrix for statistical analysis. Then the assumptions of logistic regression are checked in the following order:

1. **Approtiate outcome type**
   This is checked in cells 3 and 4.

2. **Absence of multicollinearity**
   Firstly the Variance Inflation Factor is computed for the original dataset. While some features exhibit significant multicollinearity issues, others show acceptable levels or minimal multicollinearity. The multicollinear columns are then dropped, which results in a acceptable VIF vales and reasonable correlation matrix.

3. **Linearity of independent variables**
   The Box Tidwell test only works for positive values. Hence, non-positive and null values are firstly dropped. Then the for each numeric column an additional column containing the log of the original column's values is added. Then several linear models are built and fit on various transformations of the data using the `GLM` class from the `statsmodels` library.

4. **No strongly influential outliers**
   To detect outliers Cook's distance is being calculated for each observation. Then the most influential outliers are detected and dropped.

5. **Independence of observations**
   This assumption is checked by firstly fitting a linear model to the transformed data and then graphing the residual series plot and the residual

dependence plot which do not show significant dependence of observations.

6. **Sufficiently large sample size**
   This holds, as the transformed dataset contains more than 36k samples, each only 55 variables long.

After the assumptions are checked the model's final formula is being calculated in the `step.py` file using the forward and backward elimination. Then the final model is built and fit on the transformed dataset. The model is then exported to the `model.pkl` file. Then the best cutoff can be calculated, using the `cutoff_search.ipynb` file.

4 Model properties

This section explains the code behind the examination of the model's properties (the `mode_properties.ipynb` file). We again check the assumptions of the logistic regression, now using the champion model. This process is analogous to the one described in the model development section. After that the model's ROC curve is being computed and graphed (with respect to the training data). The AUC is equal to 0.78, which we believe to be good for such a basic model. Finally several other metrics of the model's accuracy on the training dataset are being computed and displayed.

5 Model testing

This section explains the code behind the model testing process which is defined in the `model_testing.ipynb` file. Firstly, it is checked whether there was any transformation done on the target variable. After reassuring that there were none, the ROC curve is being computed and graphed, this time on the testing dataset. The area under this curve is equal to 0.76. This shows that there is no overfiting, as the diffrence between the ROC AUC's on the training and testing data is negligible. The cutoff value is set to 0.2 as after some experimenting we found it to give the best results. Several other metrics of the model's accuracy on the testing dataset are being computed and displayed.

6 Challenger model - random forest

As a challenger model we chose a model based on the random forest method. This section covers the code behind the challenger's development, defined in the `random_forest.ipynb` file. After importing libraries and the data the `roc` function is defined. This function is used to graph a given model's ROC curves on the training and testing datasets.

Several models are being tested, all developed using the `RandomForestClassifier` class from the `sklearn` library. Non-constrainting the trees' maximum depth was found to be leading to tremendous overfiting. After some experiments with the model, 2 grid searches are run in order to find the best hyperparameters. The first grid search finds the optimal values of the `n_estimators` and the `max_depth` parameters. Then a second grid search is run in order to find the best value of the `max_features` parameter with the `n_estimators` and `max_depth` fixed to the optimal values found before. Finally, a simple random forest classifier is obtained with the area under the ROC being equal to 0.72.