Coursework: Continuous Risk Evaluation of Loans

- 1. Preprocession of the data

> clean and load the data

- 2、Data Exploration

> 2.1 Similar Distributions - loan_amnt, funded_amnt and funded_amnt_inv
>
> 2.2 loan status and interest rate
>
> 2.3 Annual income
>
> 2.4 Loan status
>
> 2.5 Loan grade
>
> 2.6 Interest rate
>
> 2.7 Loan status
>
> 2.8 Interest rate

- 3. Feature Engineering

> 3.1 Drop the leak information:
>
> 3.2 multi-dimensional gap of missing values:
>
> 3.3 Drop the categories features with too many groups
>
> 3.4 Data edited:
>
> 3.5 One-hot
>
> 3.6 Train and test split

- 4. Models

> 4.1 Random forest
>
> 4.2 KNN
>
> 4.3 SVC
>
> 4.4 Logistic Regression
>
> 4.5 Gradient Boosting
>
> 4.6 Ensembling
>
> 4.7 XGboost with auto parms
>
> 4.8 Bagging Classifier
>
> 4.9 Adaptive Boosting Classifier

> 4.10 Neural Network

- 5 Time-series predictive analytics

> KNN with Classic DTW
>
> VAR models:

- 6. Conclusion

How to run the risk evaluation model(ML)

- Transform your data input as standard format:

One of the best and easier way to run to run our data cleaning steps and get the cleaned set

or, you can following the X_test format and construct the features like the same format

- As I have save the models in file, first you need load the models "gb = load('filename.joblib') " to loand the best performed models of "Gradient Boosting"
- You can get the predict "y_pred = gb.predict(X_test)"
- If you like, you can also print the varience of importance by the following code

```
gb.fit(X_train_res, y_train_res)
y_pred = gb.predict(X_test)
acc = accuracy_score(y_test,y_pred)
balanced = balanced_accuracy_score(y_test,y_pred)
recall = recall_score(y_test,y_pred,average='micro')
specificity = recall_score(y_test,y_pred, pos_label = 0,average='binary')

print('Acc. {} Recall: {} Specificity: {} Balanced:
{}'.format(acc,recall,specificity,balanced))


feature_importance = gb.feature_importances_
# make importances relative to max importance
feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5
# plt.subplot(1, 2, 2)
plt.figure(figsize=(8, 18))
plt.barh(pos, feature_importance[sorted_idx], align='center')
plt.yticks(pos, X_train.keys()[sorted_idx])
plt.xlabel('Relative Importance')
plt.title('Variable Importance')
plt.show()
```

Some table about Data cleaning steps:

Leak information table:

Column name	Desc in data dictionary	Comment
total_rec_prncp	Principal received to date	Payment only happens after the loan is issued
total_rec_int	Interest received to date	Payment only happens after the loan is issued
total_rec_late_fee	Late fees received to date	Payment only happens after the loan is issued
last_pymnt_amnt	Last total payment amount received	Payment only happens after the loan is issued
last_pymnt_d	Last month payment was received	Payment only happens after the loan is issued
recoveries	post charge off gross recovery	Only charge off loans have this
collection_recovery_fee	post charge off collection fee	Only charge off loans have this
debt_settlement_flag	None	Only charge off loans need settlement
debt_settlement_flag_date'	None	Only charge off loans need settlement
settlement_status	None	Only charge off loans need settlement
settlement_date	None	Only charge off loans need settlement
settlement_amount	None	Only charge off loans need settlement
settlement_percentage	None	Only charge off loans need settlement
settlement_term	None	Only charge off loans need settlement

- **One-hot:**

We adopt different encoding methods for different data types

· For ordinal features, it has meaning for orders, like the grade of A, B and C, we assign ordered numbers to these values, like 1,2,3 (grade, sub_grade)

· For normal categories features, like the gender of man and woman, with no meaning, we just encode those categorical features as a one-hot numeric array

A table detailing which python that was used to create each figure in the report.

This part we added into the python jupyter nootebook.