MACHINE LEARNING PROJECT 2023 – STAGE 2

Predict Loan Defaulters with Linear Regression and Random Forest Classifier

Part 1: Introduction

Finance is a big sector in automation and digitalization. Currently, many banks are developing or using machine learning and AI to minimize human labor in tasks that can be automated. This report will explore one of the most common applications: Loan defaulters. In loan defaulter, loans are granted automatically to loan that matches the standard, leaving only few to be manually managed.

This report consists of 6 parts and appendix. Part 2: Problem Formulation describes the dataset. Part 3: Methods includes feature processing and explains why two machine learning methods and loss functions are chosen. Part 4: Result compares the error between the chosen methods and Part 5: Conclusions explains which method is the better. Part 6: References includes the bibliography, and a code Appendix will be attached to the end of this document.

Part 2: Problem Formulation

This project's goal is to use machine learning to grant loans automatically when a customer enters his/her information in the loan request. The method used in this project is supervised learning with binary labels. One ("1") means that the loan is automatically granted while zero ("0") means that the loan is not.

The data set is retrieved from Klagger, originally taken from Coursera's Loan Default Prediction Challenge. [1] Each entry (or row) represents a loan request with 8 main features. The meaning and datatype of each feature are stated in Figure 1.

Feature	Explanation	Datatype
Age	The age of the borrower	integer
Income	The annual income of the borrower	integer
LoanAmount	The amount of money being borrowed	integer
CreditScore	The credit score of the borrower, indicating their creditworthiness	integer
DTIRatio	The Debt-to-income ratio, indicating the borrower's debt to their income	float
EmploymentType	The employment status of the borrower (Full-time, Part-time, Self-employed, Unemployed)	string
HasMortgage	Whether the borrower has a mortgage or not (Yes/No)	string
LoanPurpose	The purpose of the loan (Home, Auto, Education, Business, Other)	string

Figure 1. Features' explanation and datatype

Part 3: Methods

Data preprocessing

In the original data set, there were 16 features, however, to reduce the complexity of this project, only 8 features were selected. To improve the runtime of the code, the unnecessary features' columns have been removed after feature selection. Additionally, there are 3 selected features and 4 unselected features (HasCosigner, HasDependents, MaritalStatus, Education) that have string datatype. The data of these features are transformed using the label encoder for categorical feature from sklearn.preprocessing library. [2] In order to get a correlation analysis for feature selection, the 4 (future) unselected features are still processed.

Feature selection

There are 255347 entries in this data set with no null data. These features are selected based on domain knowledge. After studying the most common reasons for personal loans rejection, I have narrowed down the most important aspects when filling out loans, which are credit score, debt-to-income ratio, income (including amount and stability) and employment type. [3] Additionally, correlation analysis between these factors and the final decision shows a close connection as in figure 2.

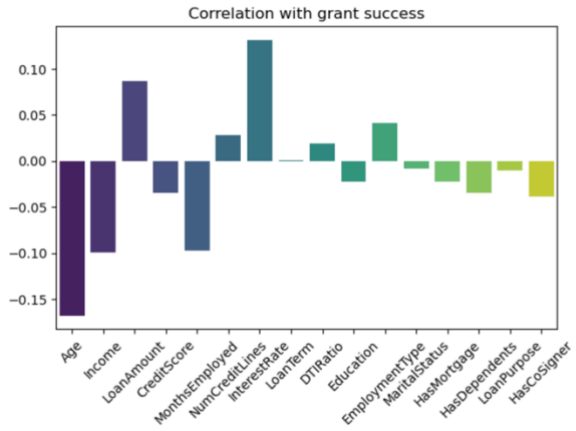


Figure 2. Correlation between the original features and grant success

Choosing the first model and loss function

Linear Regression is chosen as the first machine learning model for this problem. Even though it is not ideal for a complex problem of 8 features, it is a classic model that is worth trying. Moreover, using this function will create a good contrast with the second model (Random Forest Classifier) to demonstrate the importance of choosing the appropriate model in machine learning. The model is also available for use in sklearn.linearmodel which makes the work more convenient as this project used Python as the programming language. [2]

The mean square error loss function is used in this project since it is a classic and versatile loss function. The function is also available and ready to use in Python sklearn.metrics package. [2]

Choosing the second model and loss function

The method Random Forest Classifier was chosen for this project since it is more suitable for classification problems. It is more flexible than linear regression method when there are multiple features of different datatypes. Moreover, the method is provided in sklearn.ensemble, which makes it simple and can be used similarly to other models studied in this course. [2]

To make an accurate comparison, the mean square error loss function is also applied to Random Forest Classifier. Besides being an excellent metric in the context of optimization, the MSE is a popular classic loss function. [4] Therefore, using MSE allows this report to reach a wider range of audience.

Training and validation sets

The data is split by 80/10/10 ratio for training, validation, and testing set respectively based on the well-known Pareto principle. [5] The data set is large (more than 200 000 entries) and not sorted so the data set can be divided with the single split method and still ensures randomness. The popular K-fold method is not applied in this case because the data set is too large and repeating it kth times will cost a lot of time and computational energy.

Part 4: Result

The training and validation errors are recorded in the following table.

	Training error	Validation error
Linear Regression	0.09769559803148782	0.09703935046647708
Random Forest Classifier	5.384845087797451e-05	0.00011748580379870765

Figure 3. Training and validation errors

Observing the result, the Random Forest Classifier is the more suitable model for this machine learning problem. The errors for both training and validation for this method is significantly smaller in comparison to Linear Regression. This result is expected since the Random Forest Classifier is intentionally chosen to be more suitable than Linear Regression in this case.

The test set of the Random Forest Classifier is divided as mentioned in Part 3: Method – Training and validation test, which is 10% of the total data. The test error or the accuracy is 0.8842764832582729, which is a good result. [6] The high accuracy can also be observed from the confusion matrix.

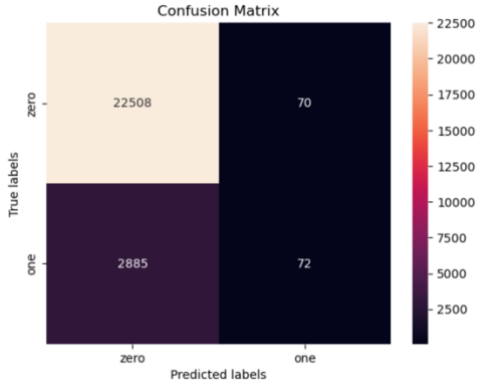


Figure 4. Confusion matrix for Random Forest Classifier model

Part 5: Conclusions

This report applies Linear Regression and Random Forest Classifier to grant loan automatically based on the borrower's age, income, and several other factors. The problem has a large data which makes it convenient when dividing the dataset for training, validation, and testing. Through applying two

"contrasting" methods, we also observe the importance of choosing the correct model for machine learning problem. The Random Forest Classifier model performs significantly better than Linear Regression model, which is suitable for more simple problem. The Random Forest Classifier model also gives an impressive result of 88.4% accuracy and the false positive result (granting loans to person who should not receive them) is unnoticeable as observed from the confusion matrix. In general, we believe that the Random Forest Classifier has a satisfactory result.

Part 6: References

[1] "Loan Default Prediction Dataset," *Kaggle*, Sep. 11, 2023. https://www.kaggle.com/datasets/nikhil1e9/loan-default

[2] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

[3] R. Safier and J. Brown, "7 reasons why your personal loan was declined (and 6 ways to fix it)," *LendingTree*, Jun. 2023, [Online]. Available: https://www.lendingtree.com/personal/reasons-why-your-personal-loan-was-declined/

[4] Zhou Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," IEEE Signal Process. Mag., vol. 26, no. 1, pp. 98–117, Jan. 2009, doi: 10.1109/MSP.2008.930649.

[5] V. R. Joseph, "Optimal ratio for data splitting," *Statistical Analysis and Data Mining*, vol. 15, no. 4, pp. 531–538, Apr. 2022, doi: 10.1002/sam.11583.

[6] A. kumar mandal and R. Sen, "Supervised Learning Methods for Bangla Web Document Categorization," International Journal of Artificial Intelligence & Applications, vol. 5, Oct. 2014, doi: 10.5121/ijaia.2014.5508.

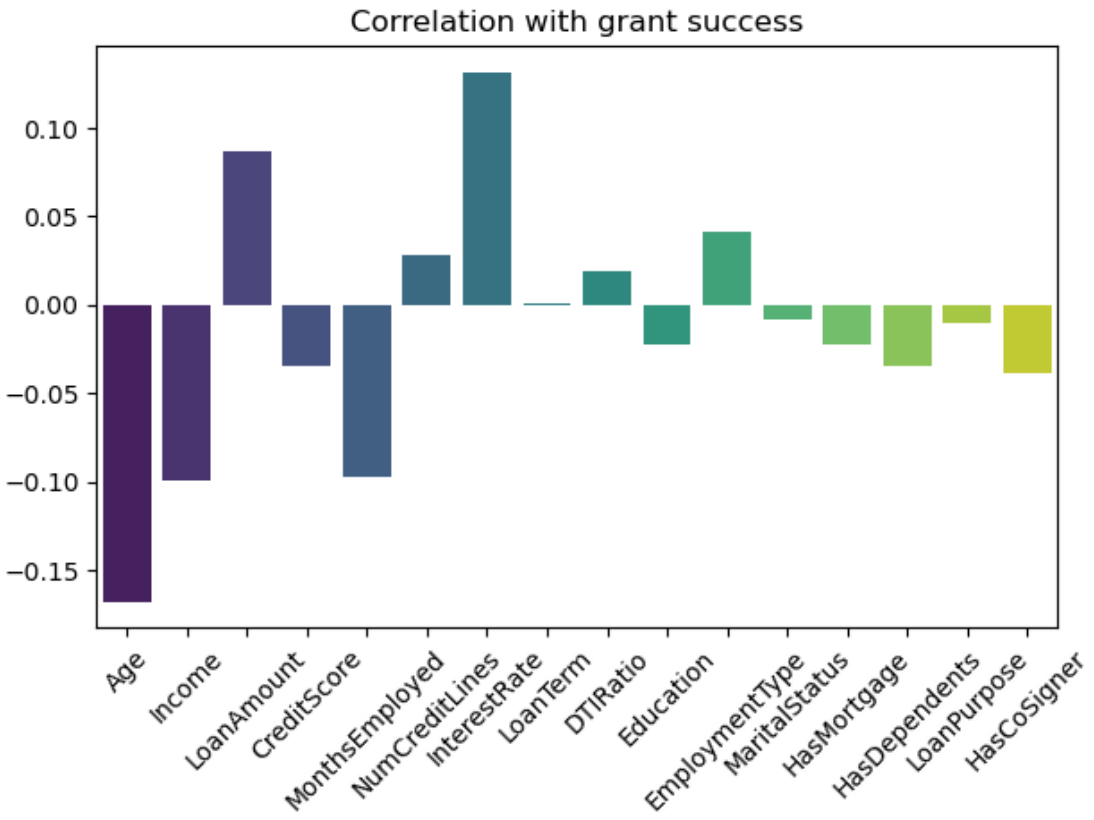
Appendix

October 9, 2023

[14]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, mean_squared_error, confusion_matrix
import seaborn as sns
```

[4]:
```
data = pd.read_csv('project.csv')
#processing string data
encoder = LabelEncoder()
cols = ['HasCoSigner','LoanPurpose','HasDependents',␣
 ↪'HasMortgage','MaritalStatus', 'EmploymentType', 'Education']
for col in cols:
    data[col] = encoder.fit_transform(data[col])

#correlation analysis
data_corr = data.drop(['LoanID','Default'],axis=1)
correlation = data_corr.corrwith(data['Default'])
sns.barplot(x=correlation.index, y=correlation.values, palette='viridis')
plt.title('Correlation with grant success')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[5]:
```
#data cleaning and selecting features
data = data.
  ↪drop(['LoanID','MonthsEmployed','NumCreditLines','InterestRate','LoanTerm','Education','Mar
data = data.dropna(axis=0)

data.sample(5)
```

[5]:

	Age	Income	LoanAmount	CreditScore	DTIRatio	EmploymentType	\
121726	40	118108	237539	822	0.69	1	
100317	51	44672	221417	835	0.17	2	
37427	24	146293	50771	670	0.53	1	
228028	59	40346	214504	520	0.25	0	
31228	44	36508	211551	318	0.36	1	

	HasMortgage	LoanPurpose	Default
121726	0	1	0
100317	0	1	0
37427	0	3	0
228028	0	1	0
31228	1	2	1

[6]:
```
X = data.drop(['Default'],axis=1)
y = data['Default'] #label if a loan is automatically granted or not
```

[7]:
```
#splitting into training and testing
X_train, X_val_test, y_train, y_val_test = train_test_split(X, y, test_size=0.
↪2, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_val_test, y_val_test,␣
↪test_size=0.5, random_state=42)
```

[9]:
```
model1 = LinearRegression()
#training
model1.fit(X_train, y_train)
y_train_pred = model1.predict(X_train)
error_train = mean_squared_error(y_train, y_train_pred)

#validation
model1.fit(X_val, y_val)
y_val_pred = model1.predict(X_val)
error_val = mean_squared_error(y_val, y_val_pred)
```

[10]:
```
print("Linear Regression result: ")
print("Error for training: ",error_train)
print("Error for validation: ",error_val)
```

```
Linear Regression result:
Error for training:  0.09769559803148782
Error for validation:  0.09703935046647708
```

[11]:
```
model2 = RandomForestClassifier()
#training
model2.fit(X_train, y_train)
y_train_pred = model2.predict(X_train)
error_train = mean_squared_error(y_train, y_train_pred)

#validation
model2.fit(X_val, y_val)
y_val_pred = model2.predict(X_val)
error_val = mean_squared_error(y_val, y_val_pred)
```

[12]:
```
print("Random Forest Classifier result: ")
print("Error for training: ",error_train)
print("Error for validation: ",error_val)
```

```
Random Forest Classifier result:
Error for training:  5.384845087797451e-05
Error for validation:  0.00011748580379870765
```

[13]:
```
#testing
y_pred_test = model2.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_test)
error_test = mean_squared_error(y_test, y_pred_test)
print("Accuracy: ", accuracy)
```

```
Accuracy:  0.8842764832582729
```

[16]:
```
#confusion matrix
conf_mat = confusion_matrix(y_test, y_pred_test)
ax= plt.subplot()

sns.heatmap(conf_mat, annot=True, fmt='g', ax=ax)

ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['zero', 'one'])
ax.yaxis.set_ticklabels(['zero', 'one'])
```

[16]:
```
[Text(0, 0.5, 'zero'), Text(0, 1.5, 'one')]
```