Report

Credit Score Classification Using Neural Network

Nguyen Thanh Vinh

Abstract—You are working as a data scientist in a global finance company. Over the years, the company has collected basic bank details and gathered a lot of credit-related information. The management wants to build an intelligent system to segregate the people into credit score brackets to reduce the manual efforts.

Keywords— $credit_score, classification$

1. Introduction

To handle the credit score classification problem, there are many way to get this problem done, like Logistic Regression , Random Forest or some other similar algorithm specify for classification. But in this time , we will use other method to solve this problem that will improve performance of problem solving process, Neural Network will be used in this time.

2. Neural Network

The `Neural␣Network` is the concept which the algorithm built to to imitate human brain, specifically how each neuron in our brain work together to solve a problem. how they work, and their applications in various fields. Understanding neural networks is essential for anyone interested in the advancements of artificial intelligence.

3. Abstract

According to attribute of Neural Network, neurons connect to each other through coefficient called `weight(w)` that will make the construction of a system has this linear form

$$a_1 w_i + a_2 w_i + 1 + ... = a_n w_(i + n)$$

(i represent for the index of weight value)

Beside of using neural network, we will use other algorithm and mathematical formula to testing the hypothesis that our model solving like `Chi_Square` , `ANOVA`, `T-statistic`, along the precess each of those method will be expose and be applied to problem solving.

4. Practical Problem Solving

4.1. Library requirement

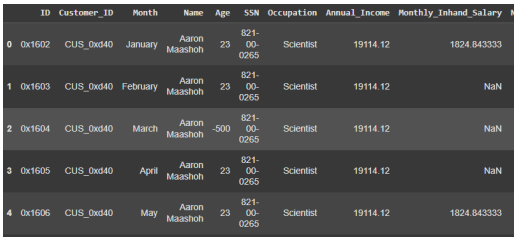
In this problem, we will use `Scikit-learn`to perform data preprocessing stage like handling mismatch value, numerical optimization, encoded labels, etc. `Tensor and Keras` to trainning and perform solving process

4.2. Module Requirement

- Scikit-learn
  - Encoding, Scale
  - Evaluation: R2_score, accuracy_scores, recall_score, precision
- Tensor and Keras
  - Dense, Dropout, BatchNormalization, Input
  - Sequential
  - Adam
  - EarlyStopping
- SMOTE

4.3. Dataset

We will use very popular `Credit score classification` dataset by Rohan Paris from Kaggle to train and test with our model: https://www.kaggle.com/datasets/parisrohan/credit-score-classification



	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.12	NaN
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333

Figure 1. Dataframe Overview

5. Information and Feature Selection

#	Column	Non-Null Count	Dtype
0	ID	30395	object
1	Customer_ID	30395	object
2	Month	30395	object
3	Name	27350	object
4	Age	30395	object
5	SSN	30395	object
6	Occupation	30395	object
7	Annual_Income	30395	object
8	Monthly_Inhand_Salary	25845	float64
9	Num_Bank_Accounts	30395	int64
10	Num_Credit_Card	30395	int64
11	Interest_Rate	30395	int64
12	Num_of_Loan	30395	object
13	Type_of_Loan	26931	object
14	Delay_from_due_date	30395	int64
15	Num_of_Delayed_Payment	28246	object
16	Changed_Credit_Limit	30395	object
17	Num_Credit_Inquiries	29801	float64
18	Credit_Mix	30394	object
19	Outstanding_Debt	30394	object
20	Credit_Utilization_Ratio	30394	float64
21	Credit_History_Age	27591	object
22	Payment_of_Min_Amount	30394	object
23	Total_EMI_per_month	30394	float64
24	Amount_invested_monthly	29046	object
25	Payment_Behaviour	30394	object
26	Monthly_Balance	29994	object
27	Credit_Score	30394	object

Table 1. Summary of Data Columns

5.1. Figures

Our dataset has total 150000 rows, but to make sure that there is no unwanted error happed in out process, the dataset was splitted to two different datasets `train.csv` and `test.csv` with radio 7:3.

Figure.1 shows us some samples taken from the dataframe, there is some missing value that can cause a bad effot to out the model tranning process, which will lead to low performance and poor accuracy.

5.2. Feature Selection and Transformation

According to machine learning theory, each coefficient has a unique weight that influences model performance. To enhance the model's effectiveness, the **Payment_Behaviour** column will be divided into **Spending_Level** and **Payment_Value**. This transformation simplifies the datatype of the original column, leading to improved model performance (eg. High, Low).

5.3. Analysis and Cleaning

There are two type of columns the dataframe contain called **Numerical Column** and **Categorical Column**. Each column type has different method to clean and handle missing values.

Median filling method is an efficient way to fill missing value, string value and and negative value will be also processed, make the dataframe more clean and improve model performance

```
    def clean_numeric_columns(self):
        for col in self.numeric_columns:
            if self.dataframe[col].dtype == 'object':
                self.dataframe[col] = self.dataframe[col
    ↪ ].str.extract(r'(\d+)')  # Extract only numeric
    ↪ digits > 0
                self.dataframe[col] = pd.to_numeric(self.
    ↪ dataframe[col], errors='coerce')  # Convert to
    ↪ number
                self.dataframe[col] = self.dataframe[col
    ↪ ].fillna(self.dataframe[col].median())  # Fill
    ↪ NaN with median
            else:
                self.dataframe[col] = self.dataframe[col
    ↪ ].fillna(self.dataframe[col].median())
```

Code 1. Numerical Columns Cleaning Process

For Categorical Column, there are some value which contain underline symbol which make non-sense so they will be removed.

```
        #Removing useless value like '_'
def remove_underscore_rows(df, column_name):

    filtered_df = df[~df[column_name].str.contains('_',
    ↪ na=False)]

    removed_rows = len(df) - len(filtered_df)
    print(f"{column_name} column removed {removed_rows}
    ↪ rows.")

    return filtered_df

for col in cat_cols:
    df = remove_underscore_rows(df, col)

```

Code 2. Catagorical Columns Cleaning

Our dataframe know is more clean and easy to do further analysis, for some row which hard to handle can be dropped if it is not a majorty in dataframe

5.4. Visualize data

Data Visualization is an important part to give us clear view about the relation between column to each other.

Besides providing plots, **heatmaps** and **correlation matrices** play an important role in identifying relationships between variables and quickly spotting strong and weak correlations.

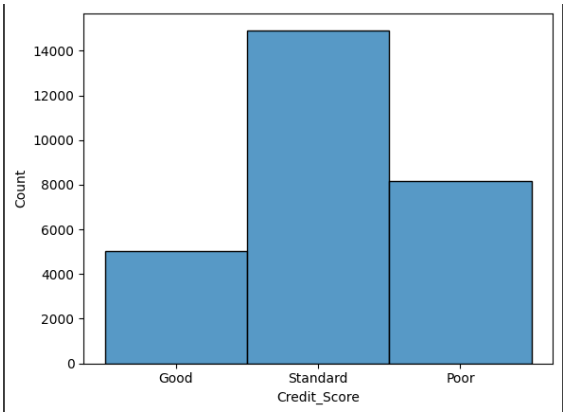
From the analysis, we can observe that **Delay From Due Date** and **Outstanding Debt** are two of the strongest correlated variables affecting the **Credit Score**. This correlation is clearly visible through the heatmap or correlation matrix.

The correlation values of **0.57** and **0.35** indicate a significant impact on the model's predictions.

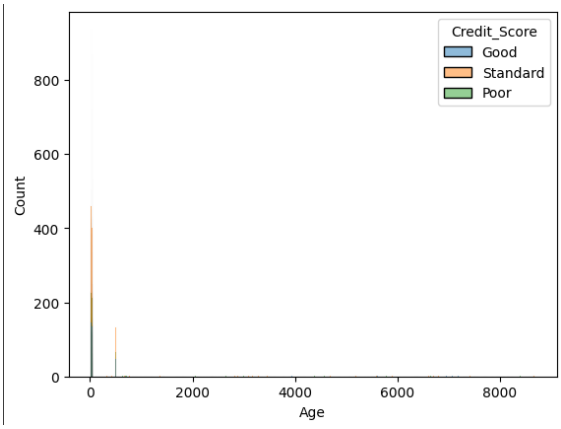
6. Hypothesis Testing

Chi-Square and F-Tests

This Python code defines a `chi_2_test` function that conducts hypothesis testing to assess the relationship between categorical and numerical columns in a DataFrame and the target variable `Credit_Score`.



(a) Prediction Unique Value



(b) Relation between Age and Credit Score

Figure 2. Data Visualization

Function Purpose: The `chi_2_test` function first prepares a copy of the DataFrame with only training data and selects categorical and numerical columns of interest. It then initializes an empty NumPy array to store test results.

The function performs two types of tests:

- **Chi-Square tests** for categorical columns.
- **F-tests (ANOVA)** for numerical columns to assess their significance in explaining `Credit_Score` variations.

Hypothesis Testing: The Chi-Square test evaluates the independence between categorical variables and

$$\chi^2 = \sum \frac{(O-E)^2}{E} \tag{1}$$

`Credit_Score`, while the F-tests assess the variance in `Credit_Score` explained by numerical variables.

$$F = \frac{(SST - SSE_U)/(K-1)}{SSE_U/(N-K)} \tag{2}$$

The results include the test statistic and p-value for each column, helping identify significant factors influencing `Credit_Score`.

After the calculation, we can see which columns have the strongest correlation to the model prediction equation, other unimportant variables will be dropped to simplify our model and prevent overfitting.

```
#Remove insignificant column

# Filter out columns with p-value > 0.1
significant_cols = chi2_summary[chi2_summary['p-value']
    ↪ <= 0.1]['column'].tolist()

```

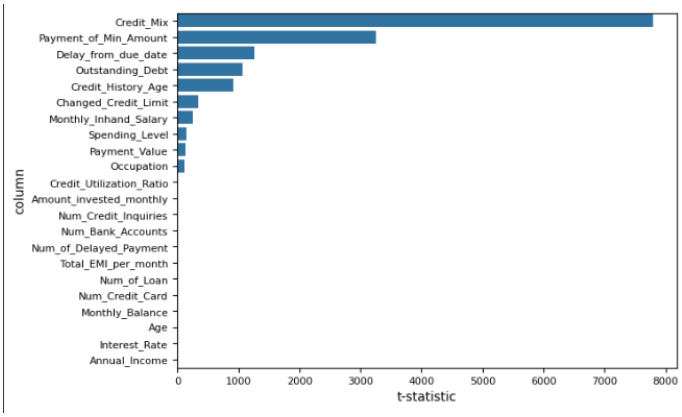


Figure 3. T-statistic Value

```
# Keep only the significant columns in your original DataFrame (df)
df = df[significant_cols + target_col]
```

Code 3. Removing unimportant variable

7. Encoding and SMOTE

7.1. Encoding Categorical Value

As we can see, **Credit Score**| unique values are "String" which cannot be used for training our model , so **Label Encoding** is a necessary step to turn `cate_col` to binary value which easy to implement to our model.

```
        one_hot_cols = ["Occupation", "Payment_of_Min_Amount"]
ordinal_cols = ["Credit_Mix", "Spending_Level", "Payment_Value"]

ordinal_categories = [
    ['Bad', 'Standard', 'Good'],
    ['Low', 'High'],
    ['Small', 'Medium', 'Large']
]

    preprocessor = ColumnTransformer(
    transformers=[
        ('num', RobustScaler(), num_cols),
        ('one_hot_enc', OneHotEncoder(handle_unknown='ignore'), one_hot_cols),
        ('ordinal_enc', OrdinalEncoder(categories=ordinal_categories, handle_unknown="use_encoded_value", unknown_value=-1), ordinal_cols)
    ]
)

```

Similarly, we classify each special columns to the right category and use special method to label or encode the value of each column

7.2. SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) is a data augmentation technique used to handle imbalanced datasets by generating synthetic samples for the minority class instead of simply duplicating existing ones.

It is part of oversampling methods and works by creating new synthetic data points along the line segments joining minority class instances in feature space.It help the model not become overfitting and improve model performance

SMOTE will take a scan through dataframe and decide which unique value should be oversampled base on percentage of value appearance.

```
    from collections import Counter
print("Before:", Counter(y_train_encoded))
smote = SMOTE(random_state=42)

X_train_resampled, y_train_resampled = smote
```

Code 4. SMOTE Implementation

Result after applying SMOTE method: Before: `Counter({np.int64(2): 6252, np.int64(1): 3664, np.int64(0): 1961})`
After: `Counter({np.int64(1): 6252, np.int64(0): 6252, np.int64(2): 6252})`

8. ANN Training

8.1. Setup for training

```
        def eval_metric(model, X_train, y_train, X_test, y_test):
    y_train_pred_probabilities = model.predict(X_train)
    y_train_pred = y_train_pred_probabilities.argmax(axis=1)
    y_pred_probabilities = model.predict(X_test)
    y_pred = y_pred_probabilities.argmax(axis=1)

    print("Test Set:")
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))

    print("\nTrain Set:")
    print(confusion_matrix(y_train, y_train_pred))
    print(classification_report(y_train, y_train_pred))

```

Split our dataset to training and testing section and, `Confusion Matix` and `Classification Report` will be also produce to summarize training result.

8.2. Epoch Training

Batch Normalization (BatchNorm) is a deep learning technique that normalizes the inputs of each layer within a neural network to stabilize and accelerate training. It helps reduce internal covariate shift, making training faster and more stable.

Batch Normalization can speed up model training process, reduces internal covariate shift, but it also got some disadvantage like can not handle big batchsize

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{4}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{5}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{6}$$

μ_B = Mean of the mini-batch.
σ_B^2 = Variance of the mini-batch.
$\hat{x}_i$ = Normalized input.
γ, β = Learnable parameters for scaling and shifting.
ϵ = Small constant to avoid division by zero.

9. Summary

```
Test Set:
[[ 304   31  156]
 [  57  571  288]
 [ 198  316 1049]]
              precision    recall  f1-score   support

           0       0.54      0.62      0.58       491
           1       0.62      0.62      0.62       916
           2       0.70      0.67      0.69      1563

    accuracy                           0.65      2970
   macro avg       0.62      0.64      0.63      2970
weighted avg       0.65      0.65      0.65      2970


Train Set:
[[5981  113  158]
 [ 133 5909  210]
 [ 229  326 5697]]
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      6252
           1       0.93      0.95      0.94      6252
           2       0.94      0.91      0.93      6252

    accuracy                           0.94     18756
   macro avg       0.94      0.94      0.94     18756
weighted avg       0.94      0.94      0.94     18756
```

Figure 4. Classification Report

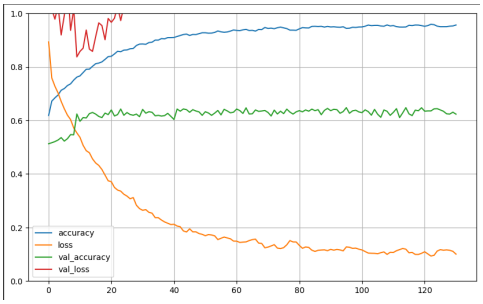


Figure 5. Training Graph

10. Contact me

You can contact me through these methods.

https://www.facebook.com/profile.php?id=100075320748765locale=vi$_V N$
vinhnt.23bi14453@usth.edu.vnb
No. 0947456078

11. Reference

`Kaggle by Yaşar Yiğit Turan` : https://www.kaggle.com/code/yaaryiitturan/credit-score-prediction-using-ann-smote/notebook

`Investopia`: https://www.investopedia.com/terms/c/chi-square-statistic.asp