# COMP4434 Big Data Analytics
# Individual Project

Report on applying machine learning and deep learning models on loans amount regression problem

by: WONG CHUN WAH

SID: 20017388D

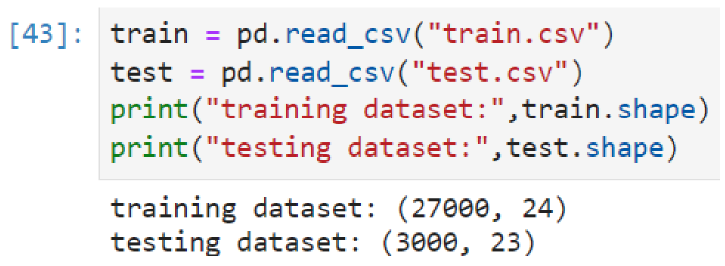
## Content:

## Introduction:

As the artificial intelligence generation has come, more and more AI applications appear in many kinds of industry. The AI application can produce huge potential amounts that helps to replace the traditional operation system. With such powerful AI techniques, it not only helps the business to make the best decision but also save the plenty of human resources. AlphaMoney , one of the famous platforms for providing loan service to clients. The platform database stored the past client personal data. In this report, the aim is to assist AlphaMoney to predict the loan amounts from the new users given their personal data. All the data processing procedure would be conducted by python programming language. The machine learning and deep learning techniques would be applied to build up the predictive model. After some model analysis and comparison, the best model would be selected, and it would become the generalized loan amount predictive model for AlphaMoney.

## Data preprocessing/analytics

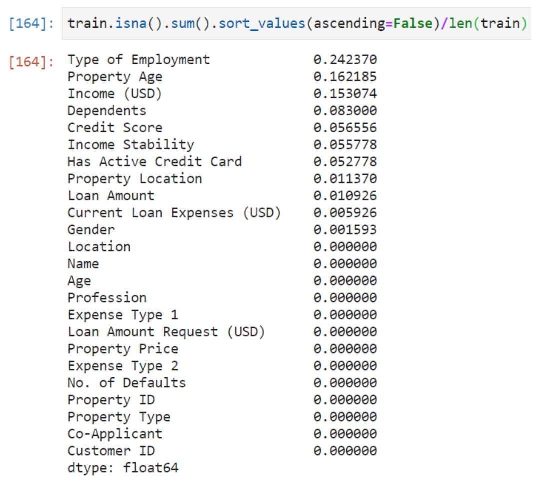
Before dive into the predictive model building section, it is more important to understand the data environment in the user dataset. Let's check the data size first, with using the python codes, we have:



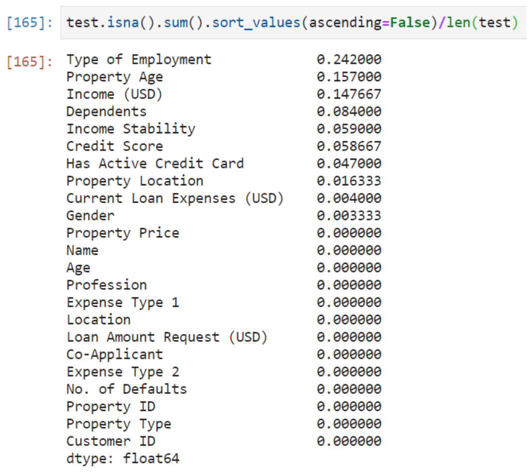
```
[43]: train = pd.read_csv("train.csv")
      test = pd.read_csv("test.csv")
      print("training dataset:",train.shape)
      print("testing dataset:",test.shape)

      training dataset: (27000, 24)
      testing dataset: (3000, 23)
```

So, we have 27000 clients records with 23 features variable and 1 target variable which is loan amount. This would be our training dataset for developing the predictive models. On the other hand, we have 3000 new clients of those loan amounts need to be predicted. That would be our testing dataset.

After understanding the data size, it is also important to understand the dataset integrity. That's how is the cases of data missing and count how many records are missing in each feature columns in both two datasets. So, the following cases shown as below:



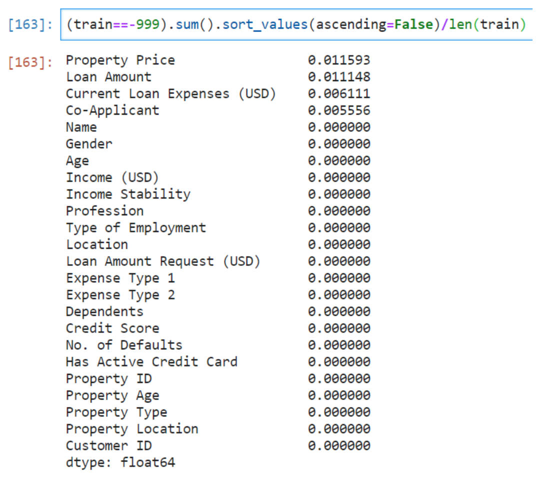
```
[164]: train.isna().sum().sort_values(ascending=False)/len(train)

[164]: Type of Employment            0.242370
       Property Age                  0.162185
       Income (USD)                  0.153074
       Dependents                    0.083000
       Credit Score                  0.056556
       Income Stability              0.055778
       Has Active Credit Card        0.052778
       Property Location             0.011370
       Loan Amount                   0.010926
       Current Loan Expenses (USD)   0.005926
       Gender                        0.001593
       Location                      0.000000
       Name                          0.000000
       Age                           0.000000
       Profession                    0.000000
       Expense Type 1                0.000000
       Loan Amount Request (USD)     0.000000
       Property Price                0.000000
       Expense Type 2                0.000000
       No. of Defaults               0.000000
       Property ID                   0.000000
       Property Type                 0.000000
       Co-Applicant                  0.000000
       Customer ID                   0.000000
       dtype: float64
```



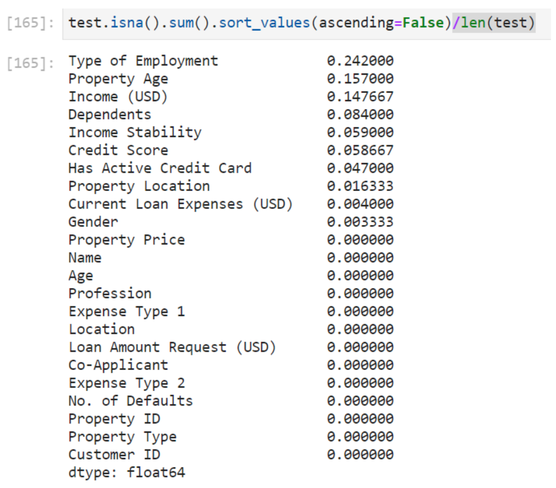
```
[165]: test.isna().sum().sort_values(ascending=False)/len(test)

[165]: Type of Employment            0.242000
       Property Age                  0.157000
       Income (USD)                  0.147667
       Dependents                    0.084000
       Income Stability              0.059000
       Credit Score                  0.058667
       Has Active Credit Card        0.047000
       Property Location             0.016333
       Current Loan Expenses (USD)   0.004000
       Gender                        0.003333
       Property Price                0.000000
       Name                          0.000000
       Age                           0.000000
       Profession                    0.000000
       Expense Type 1                0.000000
       Location                      0.000000
       Loan Amount Request (USD)     0.000000
       Co-Applicant                  0.000000
       Expense Type 2                0.000000
       No. of Defaults               0.000000
       Property ID                   0.000000
       Property Type                 0.000000
       Customer ID                   0.000000
       dtype: float64
```

All the values are in percentage form. It is clearly to see that both of dataset data missing cases are quite serious. Therefore, the data missing problem needs to be dealt with first.

Moreover, since the outliers can be also highly affecting the subsequent model accuracy, let’s see the situation of outlier’s frequency in each columns (treated as -999 value) in both two datasets:



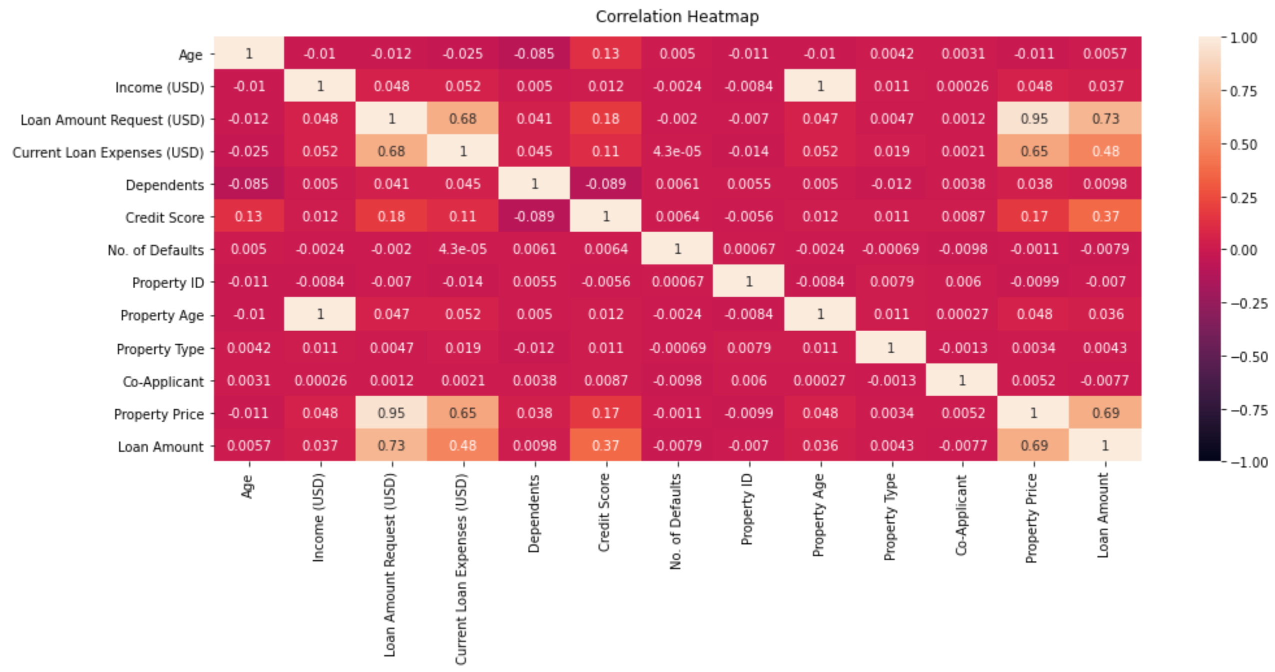
```
[163]: (train==-999).sum().sort_values(ascending=False)/len(train)

[163]: Property Price                  0.011593
       Loan Amount                     0.011148
       Current Loan Expenses (USD)     0.006111
       Co-Applicant                    0.005556
       Name                            0.000000
       Gender                          0.000000
       Age                             0.000000
       Income (USD)                    0.000000
       Income Stability                0.000000
       Profession                      0.000000
       Type of Employment              0.000000
       Location                        0.000000
       Loan Amount Request (USD)       0.000000
       Expense Type 1                  0.000000
       Expense Type 2                  0.000000
       Dependents                      0.000000
       Credit Score                    0.000000
       No. of Defaults                 0.000000
       Has Active Credit Card          0.000000
       Property ID                     0.000000
       Property Age                    0.000000
       Property Type                   0.000000
       Property Location               0.000000
       Customer ID                     0.000000
       dtype: float64
```



```
[165]: test.isna().sum().sort_values(ascending=False)/len(test)

[165]: Type of Employment              0.242000
       Property Age                    0.157000
       Income (USD)                    0.147667
       Dependents                      0.084000
       Income Stability                0.059000
       Credit Score                    0.058667
       Has Active Credit Card          0.047000
       Property Location               0.016333
       Current Loan Expenses (USD)     0.004000
       Gender                          0.003333
       Property Price                  0.000000
       Name                            0.000000
       Age                             0.000000
       Profession                      0.000000
       Expense Type 1                  0.000000
       Location                        0.000000
       Loan Amount Request (USD)       0.000000
       Co-Applicant                    0.000000
       Expense Type 2                  0.000000
       No. of Defaults                 0.000000
       Property ID                     0.000000
       Property Type                   0.000000
       Customer ID                     0.000000
       dtype: float64
```

Fortunately, the outliers’ cases are not that serious. We are now going to pre-process the data and then review the features.

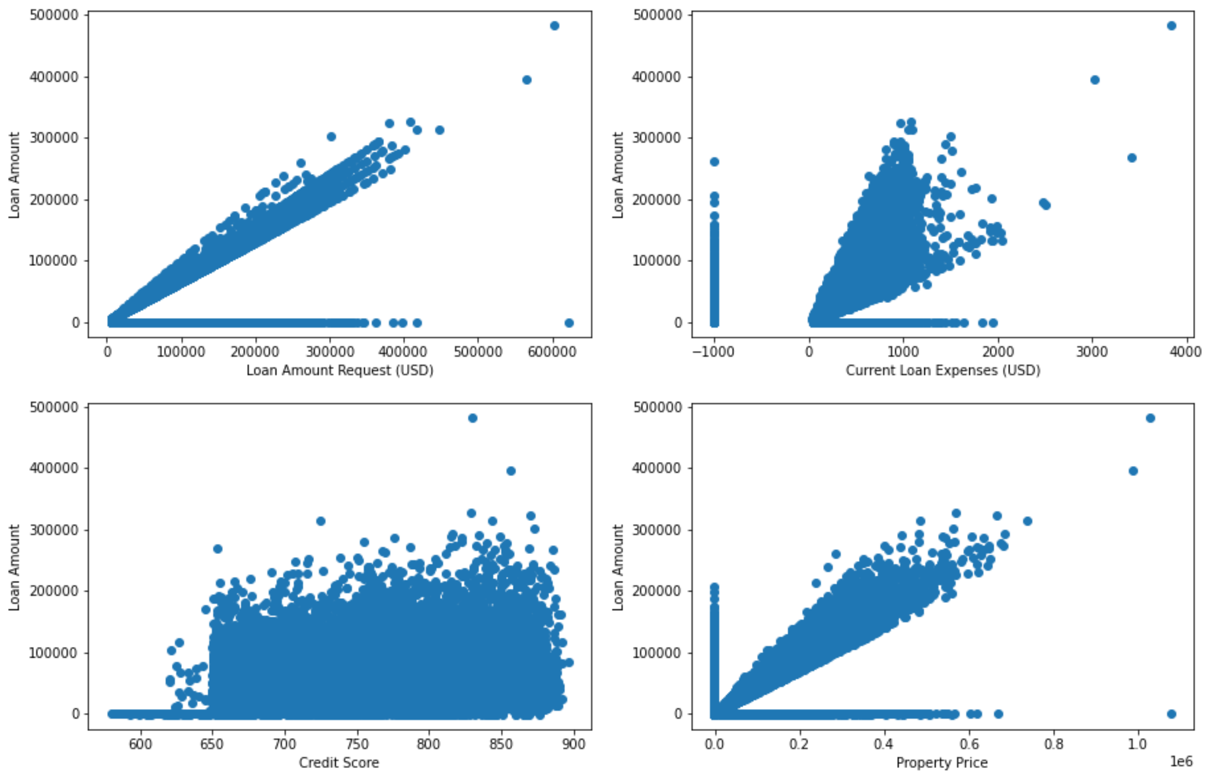
For the first two features which are “Customer_ID” and “Name”, these two would not be considered as basically they don’t affect the loan amount. Next, let’s see the correlation matrix in order to understand the linear relationships. The matrix shows as follows:



| | Age | Income (USD) | Loan Amount Request (USD) | Current Loan Expenses (USD) | Dependents | Credit Score | No. of Defaults | Property ID | Property Age | Property Type | Co-Applicant | Property Price | Loan Amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Age | 1 | -0.01 | -0.012 | -0.025 | -0.085 | 0.13 | 0.005 | -0.011 | -0.01 | 0.0042 | 0.0031 | -0.011 | 0.0057 |
| Income (USD) | -0.01 | 1 | 0.048 | 0.052 | 0.005 | 0.012 | -0.0024 | -0.0084 | 1 | 0.011 | 0.00026 | 0.048 | 0.037 |
| Loan Amount Request (USD) | -0.012 | 0.048 | 1 | 0.68 | 0.041 | 0.18 | -0.002 | -0.007 | 0.047 | 0.0047 | 0.0012 | 0.95 | 0.73 |
| Current Loan Expenses (USD) | -0.025 | 0.052 | 0.68 | 1 | 0.045 | 0.11 | 4.3e-05 | -0.014 | 0.052 | 0.019 | 0.0021 | 0.65 | 0.48 |
| Dependents | -0.085 | 0.005 | 0.041 | 0.045 | 1 | -0.089 | 0.0061 | 0.0055 | 0.005 | -0.012 | 0.0038 | 0.038 | 0.0098 |
| Credit Score | 0.13 | 0.012 | 0.18 | 0.11 | -0.089 | 1 | 0.0064 | -0.0056 | 0.012 | 0.011 | 0.0087 | 0.17 | 0.37 |
| No. of Defaults | 0.005 | -0.0024 | -0.002 | 4.3e-05 | 0.0061 | 0.0064 | 1 | 0.00067 | -0.0024 | -0.00069 | -0.0098 | -0.0011 | -0.0079 |
| Property ID | -0.011 | -0.0084 | -0.007 | -0.014 | 0.0055 | -0.0056 | 0.00067 | 1 | -0.0084 | 0.0079 | 0.006 | -0.0099 | -0.007 |
| Property Age | -0.01 | 1 | 0.047 | 0.052 | 0.005 | 0.012 | -0.0024 | -0.0084 | 1 | 0.011 | 0.00027 | 0.048 | 0.036 |
| Property Type | 0.0042 | 0.011 | 0.0047 | 0.019 | -0.012 | 0.011 | -0.00069 | 0.0079 | 0.011 | 1 | -0.0013 | 0.0034 | 0.0043 |
| Co-Applicant | 0.0031 | 0.00026 | 0.0012 | 0.0021 | 0.0038 | 0.0087 | -0.0098 | 0.006 | 0.00027 | -0.0013 | 1 | 0.0052 | -0.0077 |
| Property Price | -0.011 | 0.048 | 0.95 | 0.65 | 0.038 | 0.17 | -0.0011 | -0.0099 | 0.048 | 0.0034 | 0.0052 | 1 | 0.69 |
| Loan Amount | 0.0057 | 0.037 | 0.73 | 0.48 | 0.0098 | 0.37 | -0.0079 | -0.007 | 0.036 | 0.0043 | -0.0077 | 0.69 | 1 |

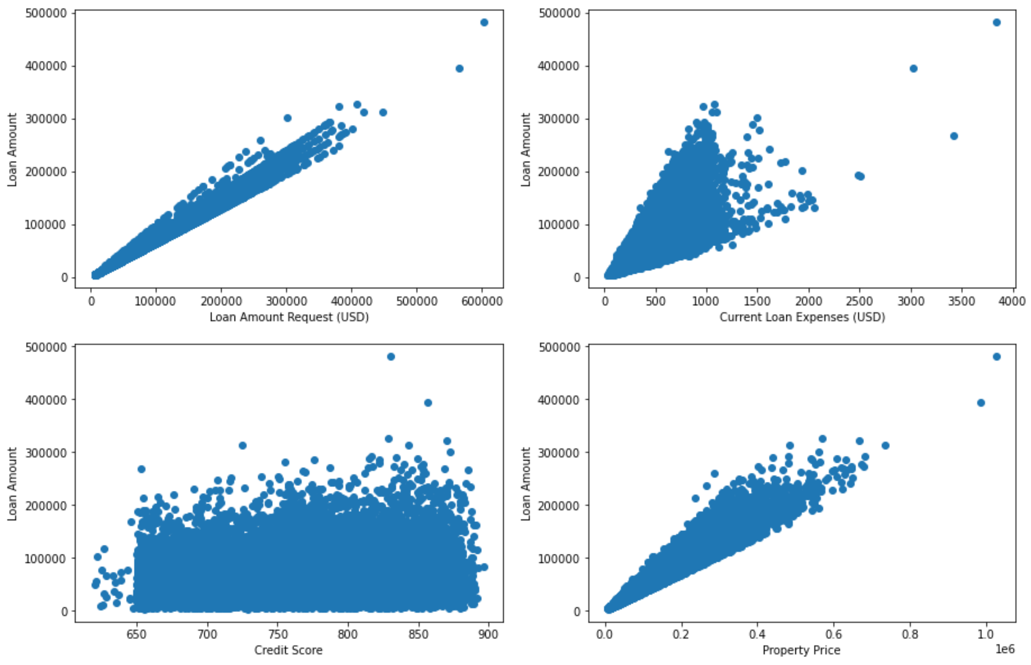
As shown from the matrix, with focusing on the dependent variable, Loan Amount, it is clearly to see that there are four independent variables including Loan Amount Request (USD), Current Loan Expenses (USD), Credit Score and Property Price appearing the strong linear relationship. Therefore, we would like to firstly preprocess these four variables.

Because of the reason of outliers, we should understand its distribution in different features. The scatter plots can concisely generate this kind of information.



The scatter plots showing the outliers normally distributed as a horizontal or vertical line. For example, we can see that the Loan Amount regarding the Current Loan Expenses (USD) scatter plot, a bunch of outlier's stacks at the -999 of x-axis. Those values should be removed before using for building model. These four-plotting showing the positive linear relationship and it would be using to build up the linear regression model.

The outliers could be found with using the quantile of data. After updating the data, the outliers in the training dataset are removed, the situation becomes:



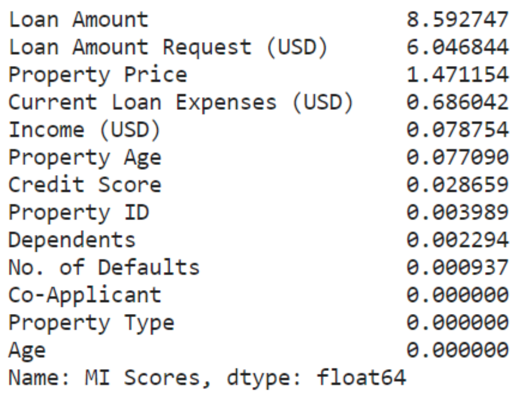
```
Training size: (18940, 22)
```

The distributions would be perfectly fit to train the linear regression model and other model as the pairs of features appears the obvious linear relationships. The numbers of existing data after cleaning the outliers is 18940. Based on the strong association result in correlation statistics, these four features are selected to be the model input features for doing the prediction of loan amount

## Mutual information measurement on features

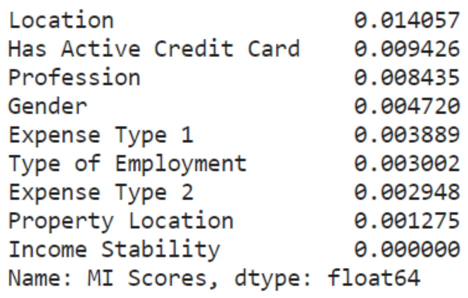
Mutual information is a powerful measurement in machine learning that assists to reveal the complex relationship between the features and target variable. Comparing with correlation, it is often be used to check the non-linear relationship. One of the advantages of using mutual information in selecting the features is mutual information can be available on calculating on both numerical and categorical data format. This helps us to discover the categorical features of importance regarding to the loan amount. Let’s see the mutual information score on both categorical and numerical features:

For numerical features:



```
Loan Amount                   8.592747
Loan Amount Request (USD)     6.046844
Property Price                1.471154
Current Loan Expenses (USD)   0.686042
Income (USD)                  0.078754
Property Age                  0.077090
Credit Score                  0.028659
Property ID                   0.003989
Dependents                    0.002294
No. of Defaults               0.000937
Co-Applicant                  0.000000
Property Type                 0.000000
Age                           0.000000
Name: MI Scores, dtype: float64
```

For categorical features:



```
Location                 0.014057
Has Active Credit Card   0.009426
Profession               0.008435
Gender                   0.004720
Expense Type 1           0.003889
Type of Employment       0.003002
Expense Type 2           0.002948
Property Location        0.001275
Income Stability         0.000000
Name: MI Scores, dtype: float64
```

Based on the result, the mutual information score of numerical features are largely higher than the mutual information score of categorical features and the score in categorical features are relatively too low. Therefore, we won’t consider the categorical features for input data in the subsequent training model. For numerical features, basically the result of top strong relation features in mutual information are the same with the result of correlation except the feature “Credit Score”. However, we can still consider including it in the features selection section.

## Model design and implementation

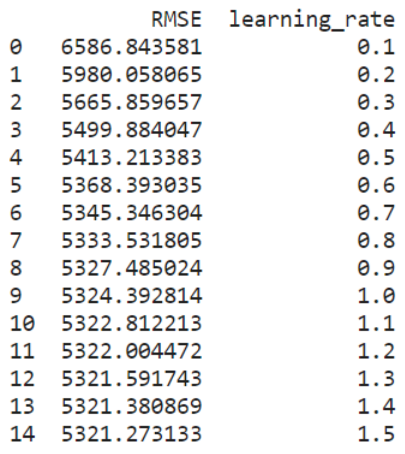
After cleaning the dataset and the features selection sections, the machine learning models are ready to do the training for predicting the loan amount using the four features including “Loan Amount Request (USD), Current Loan Expenses (USD), Credit Score and Property Price”. In this project, the linear regression and ridge regression models would be used, and its methodology of training is gradient descent. That’s, it would involve parameter tuning for choosing the optimal parameter in order to achieve the convergence. To achieve the convergence faster, the Min-Max normalization would be adopted in the feature scaling. What’s more, other regression models also being considered in this project such as decision tree regressor, random forest regressor, K-Neighbors regressor, extra tree regressor, support vector regressor and XGBoost regressor. Through adding some regression models, there could be some interesting comparison between them and benefit to select the best model to generalize the regression model. Meanwhile, the linear regression and ridge regression models would be the focus topic of parameters tuning.

## Performance evaluation and discussions

### **Linear regression**

Linear regression is one of the popular regression models to predict the numerical amounts. Linear regression model benefits to handle the linear relationship prediction and its structure is easy to understand. Comparing other models, linear regression model involves less theoretical concept and so we could be easy to implement it. For the cases, the gradient descent method would be adopted, and the learning rate becomes the only one parameter tuning. As there are four feature input values, including the intercept, totally five weighting values exists in linear regressions models. In order to ensure the convergence is attained, the 10000 iterations fixed for updating the weighting values.

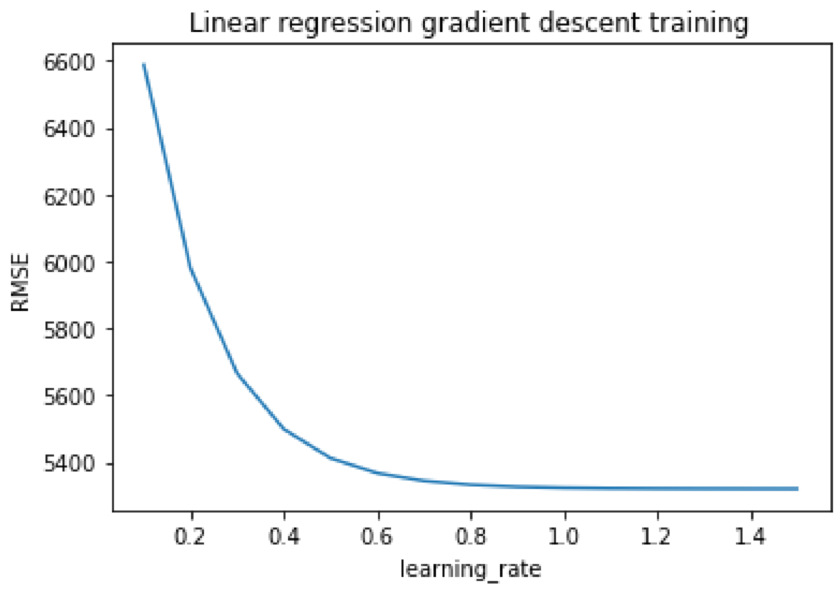
The learning rate of gradient descent controls the algorithms learning speed and its purpose is to achieve the local minimum to solve the optimization problems. At the same time, it minimizes the cost function. However, recall the aims of project, we would focus on minimizing the RMSE more than the cost function. For obtaining the optimal learning rate value, the initializations of learning rate is the most important as it gives the information on the range of optimal value. Assume the learning rate is set from 0.1 to 1.5, by using python, the corresponding RMSE would be shown as follows:



```
           RMSE  learning_rate
0   6586.843581            0.1
1   5980.058065            0.2
2   5665.859657            0.3
3   5499.884047            0.4
4   5413.213383            0.5
5   5368.393035            0.6
6   5345.346304            0.7
7   5333.531805            0.8
8   5327.485024            0.9
9   5324.392814            1.0
10  5322.812213            1.1
11  5322.004472            1.2
12  5321.591743            1.3
13  5321.380869            1.4
14  5321.273133            1.5
```

By observing the learning rate and RMSE, clearly the results showed that the larger of the learning rate, the smaller of the RMSE.

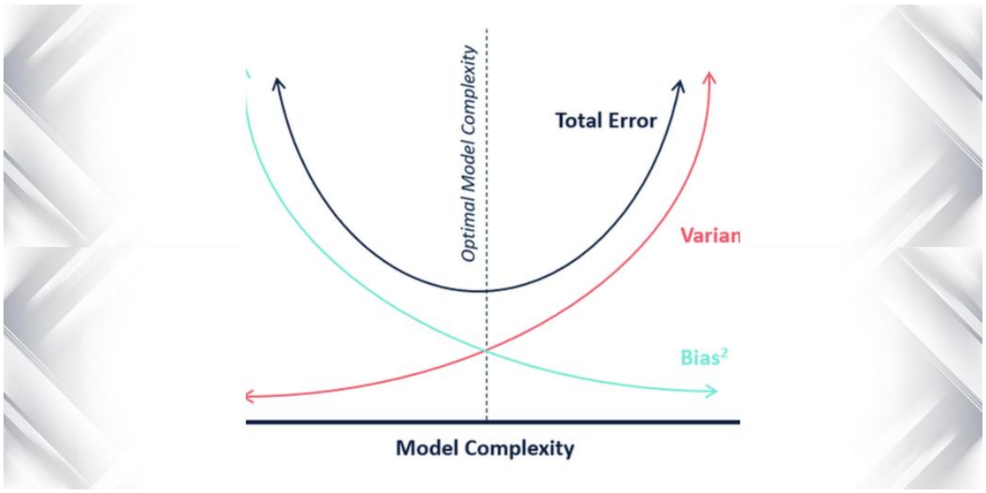
Representing in the curve graph, the following trend would become:



Based on the curve trend and the table result, the local minimum is achieved when the RMSE equal to around 5321. We may conclude that the linear regression in gradient descent is having the optimal value when the RMSE = 5321 and learning rate = 1.3.

## Ridge regression

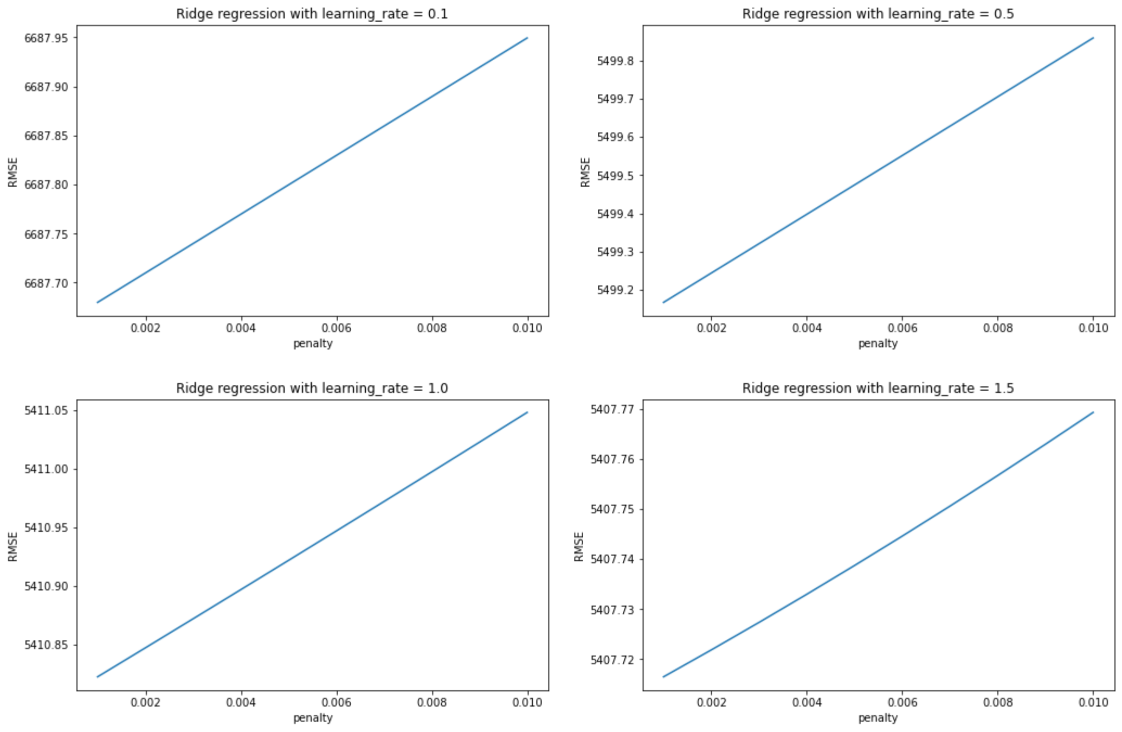
Ridge regression is a regularization technique that assists linear regression solving the overfitting problems. In view of the cases, there are totally four feature inputs. The multiple linear regression model is hence used. However, it may not have satisfying result because the independent variables could be highly correlated. This is called multicollinearity. The ridge regression would make the coefficient estimators becomes biased and the main reason is the coefficients would not be estimated by the ordinary least squares. For the advantages, the variance would be decreased. That's the Bias-variance trade off.



sourced from Wikipedia

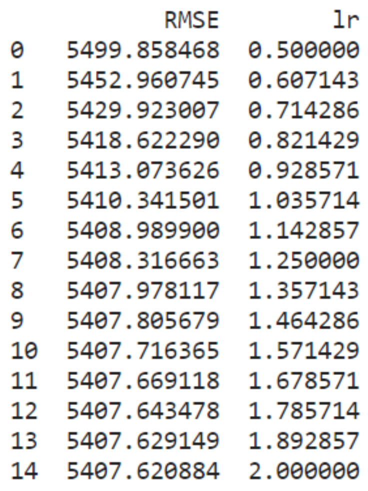
Depends on the model complexity, the ideal case is to choose the low variance and low bias concurrently. Both metrics would affect the predictive result relatively.

Again, the gradient descent is considered to find out the best parameter values in ridge regression. As a result, there are two tuning parameters which are learning rate and penalty term. The nature of learning rate is the same in the structure of gradient descent of linear regression while the penalty term is to regularize the regression model for shrinking the coefficients. The model would become ordinary linear squares if the penalty is 0 while the coefficients of model would tend to be smaller if the penalty is too large. To discover such characteristics in our purpose, let's check the penalty term first. Assume setting the penalty term from 0.001 to 0.01 and assigning four learning rate which are 0.1, 0.5, 1.0 and 1.5. The following trends are shown below:

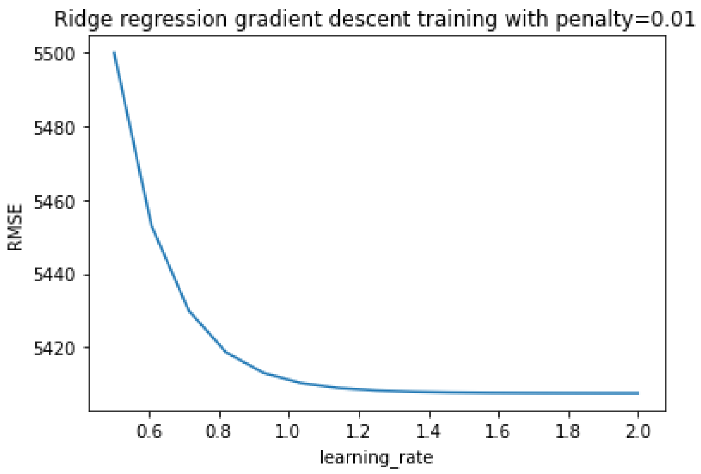


Obviously, according to these four different learning rates of graphs, the RMSE decreases if the penalty is set to be lower. Simply sum up, the penalty term does not actually matter to decrease the RMSE as we can see the changes of RMSE is generally small when penalty decreases. Therefore, the penalty term is shortly chosen as 0.01 for subsequent learning rate tuning.

Regarding to the learning rate tuning in ridge regression, the training method doing the same with linear regression. That is, the learning rate ranges from 0.5 to 2 and its training performance illustrated as follows:



```
           RMSE        lr
0   5499.858468  0.500000
1   5452.960745  0.607143
2   5429.923007  0.714286
3   5418.622290  0.821429
4   5413.073626  0.928571
5   5410.341501  1.035714
6   5408.989900  1.142857
7   5408.316663  1.250000
8   5407.978117  1.357143
9   5407.805679  1.464286
10  5407.716365  1.571429
11  5407.669118  1.678571
12  5407.643478  1.785714
13  5407.629149  1.892857
14  5407.620884  2.000000
```



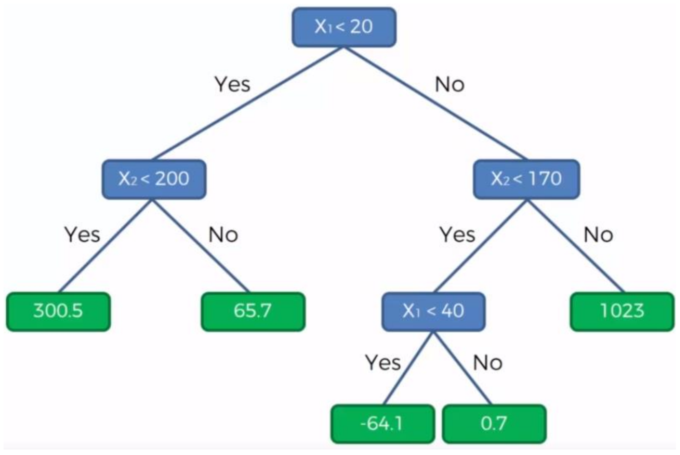
The learning curve dived rapidly from 0.5 of learning rate and decreasing become slower started at 1.0 of learning rate which is similar to previous gradient descent of linear regression. As a result, the smallest RMSE in ridge regression is 5407 by observing.

## Other regression models:

In this project, beyond linear regression and ridge regression, other regression model such as decision tree regressor, random forest regressor, K-neighbors regressor, extra tree regressor, support vector regressor and XGBoost regressor, considered to carry out the regression performance analysis. As each of those models could be the huge topic, the explanations of its structure would be short.

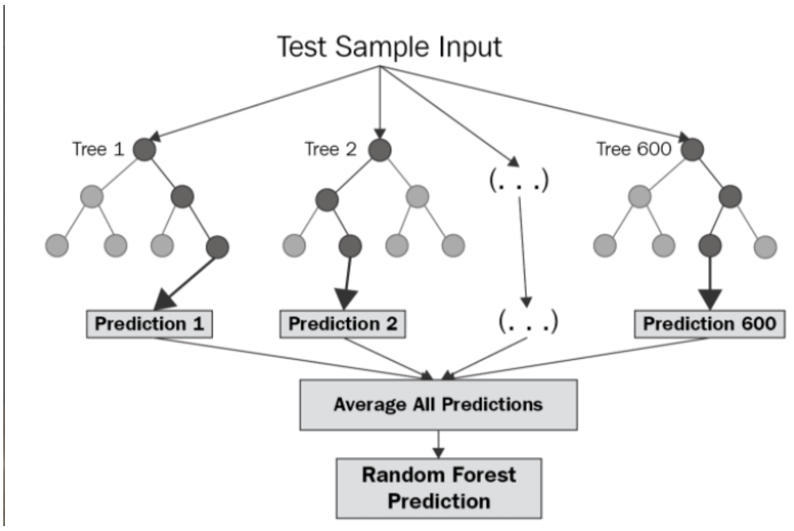
decision tree regressor:

In general, decision tree in machine learning apply in classification problems. In regression problems, it could be using the to divide the attributes value under the tree nodes and branches. Some regression metrics such as RMSE, coefficient of determination, would be also available for decision tree to evaluate the regression performance. The decision tree diagram describes as below:



random forest regressor:

Similar to decision tree, random forest regressor is a tree-based model. The main difference is random forest could be do the sampling among the dataset and generate couples of small decision tree models to do the prediction. All the subset of decision tree models generating result would affect the random forest model as random forest model make the final prediction using the majority rules of all small decision tree models. Normally, random forest could prevent the overfitting problem as the numbers of tree are small.



K-neighbors regressor:

The K-neighbors model is the simplest model structure in machine learning. The main idea of K-neighbors model is to predict the new data using the K numbers of closest existing data value of distance. It is easy to understand and doesn’t involve too many concepts. In regression, the K-neighbor regressor would find out the K close related data value and calculate the mean value among those related data value for doing prediction extra tree regressor:

The logic of extra tree regression model is mostly the same with random forest regressor. The main difference between them is extra tree model would separate the dataset randomly into the sub-samples and take the average of those decision tree models while random forest would select nodes splitting optimally to represent the data.

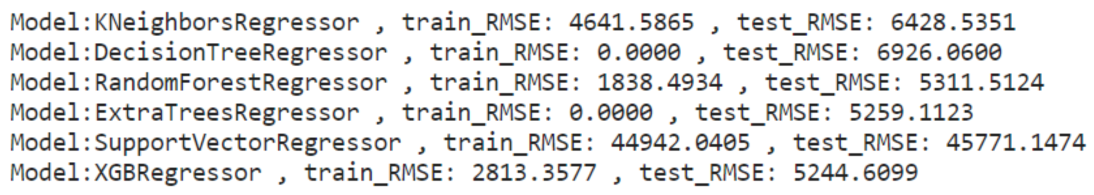
support vector regressor:

In classification, support vector machine is to develop a decision boundary which is called hyperplane to do the classification. The hyperplane aims to maximizes the margin between the classes. In regression, support vector machine merely adopt the hyperplane as a regression model.

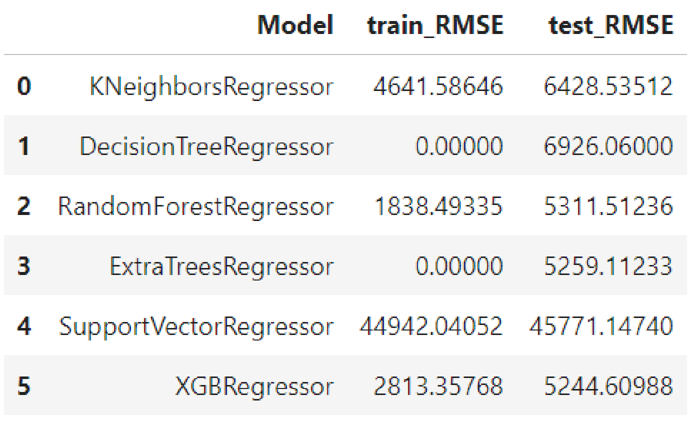
XGBoost regressor:

XGBoost regressor one of the popular ensembles learning method and distributed gradient boosting tree-based model. The powerful of XGBoost is its model leverage the fast execution speed to do the prediction. XGBoost implements the gradient descent through minimizing the convex loss function. The model combines the L1 and L2 regularization techniques.

To evaluate those machine learning models, the dataset splitting use on the given training dataset in order to understand each of the regression models performance. The training dataset and testing dataset would be divided to the whole given dataset of 80% and 20% respectively. The unseen data section performance is the main evaluation to reflect how the model perform well and the RMSE still become the only one regression metric.



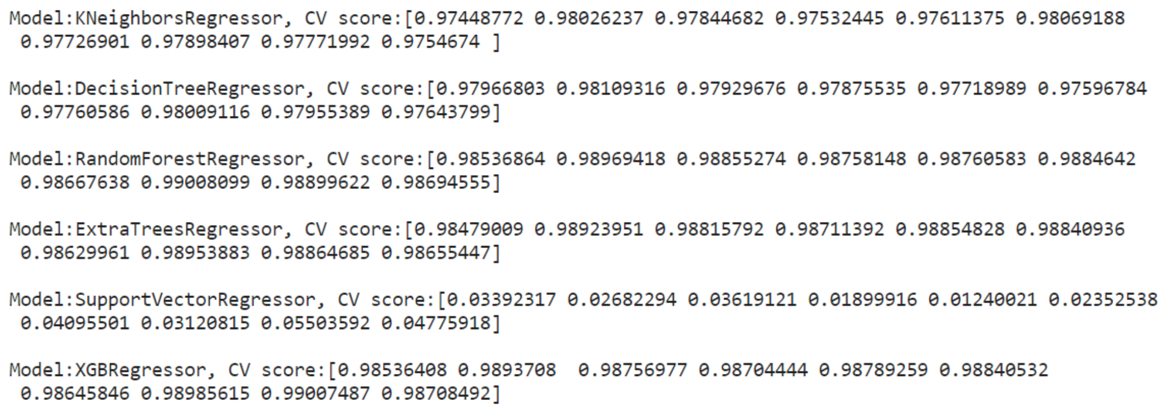
```
Model:KNeighborsRegressor , train_RMSE: 4641.5865 , test_RMSE: 6428.5351
Model:DecisionTreeRegressor , train_RMSE: 0.0000 , test_RMSE: 6926.0600
Model:RandomForestRegressor , train_RMSE: 1838.4934 , test_RMSE: 5311.5124
Model:ExtraTreesRegressor , train_RMSE: 0.0000 , test_RMSE: 5259.1123
Model:SupportVectorRegressor , train_RMSE: 44942.0405 , test_RMSE: 45771.1474
Model:XGBRegressor , train_RMSE: 2813.3577 , test_RMSE: 5244.6099
```



|  | Model | train_RMSE | test_RMSE |
|---|---|---|---|
| **0** | KNeighborsRegressor | 4641.58646 | 6428.53512 |
| **1** | DecisionTreeRegressor | 0.00000 | 6926.06000 |
| **2** | RandomForestRegressor | 1838.49335 | 5311.51236 |
| **3** | ExtraTreesRegressor | 0.00000 | 5259.11233 |
| **4** | SupportVectorRegressor | 44942.04052 | 45771.14740 |
| **5** | XGBRegressor | 2813.35768 | 5244.60988 |

Based on the result from table, the tree-based model would have a better prediction result. In practice, comparing with the linear and ridge regression model, the RMSE are similar. Although the models performed very well in training dataset, there is still having a huge result difference from the testing dataset.

There could be having some improvements of each of the model. Throughout the generalization of model is always the most concern place. The cross validation assists the model how to separate the training and testing dataset properly for improving the accuracy. The cross validation of score-based evaluation in python could inform the key information of splitting the dataset effectively. Setting the cross valid equals to 10, it means the K-folds would be 10 to do the re-sampling in each of the regression models.



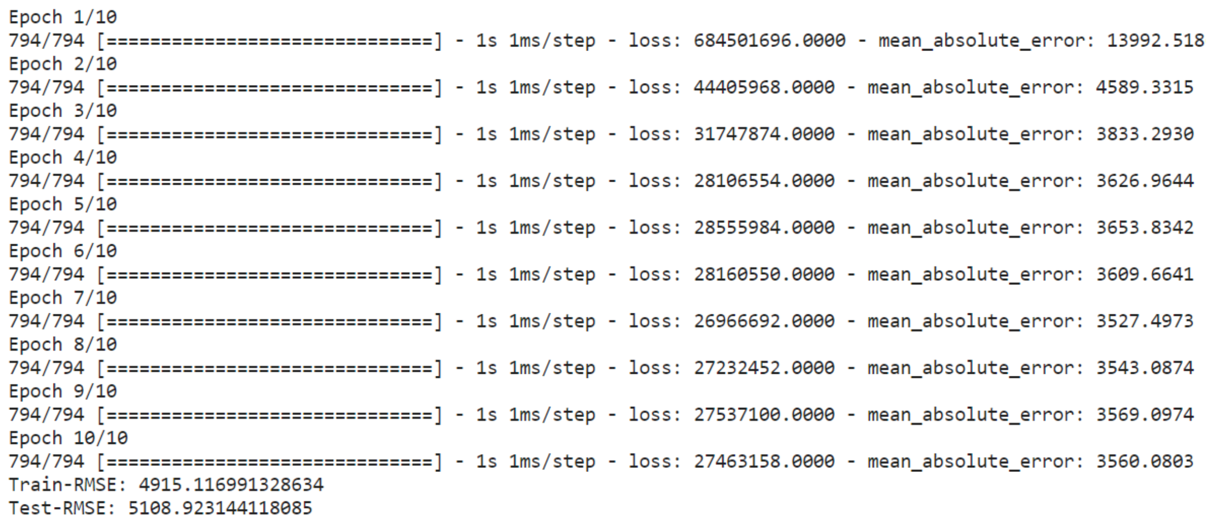
```
Model:KNeighborsRegressor, CV score:[0.97448772 0.98026237 0.97844682 0.97532445 0.97611375 0.98069188
 0.97726901 0.97898407 0.97771992 0.9754674 ]

Model:DecisionTreeRegressor, CV score:[0.97966803 0.98109316 0.97929676 0.97875535 0.97718989 0.97596784
 0.97760586 0.98009116 0.97955389 0.97643799]

Model:RandomForestRegressor, CV score:[0.98536864 0.98969418 0.98855274 0.98758148 0.98760583 0.9884642
 0.98667638 0.99008099 0.98899622 0.98694555]

Model:ExtraTreesRegressor, CV score:[0.98479009 0.98923951 0.98815792 0.98711392 0.98854828 0.98840936
 0.98629961 0.98953883 0.98864685 0.98655447]

Model:SupportVectorRegressor, CV score:[0.03392317 0.02682294 0.03619121 0.01899916 0.01240021 0.02352538
 0.04095501 0.03120815 0.05503592 0.04775918]

Model:XGBRegressor, CV score:[0.98536408 0.9893708  0.98756977 0.98704444 0.98789259 0.98840532
 0.98645846 0.98985615 0.99007487 0.98708492]
```

According to the CV score in each of regression models, the scores are almost the same regarding to its models. It is to believe that the cross validation applying in each model comes with the small effect on improving the predictive accuracy.

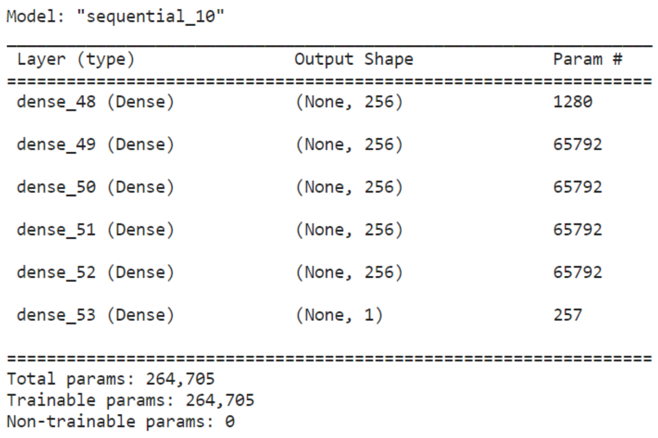
## Deep neural network

The DNN is the powerful prediction structure. It is a deep learning field in AI and the main task of DNN is to deal with the very complex prediction problem and large space for development. In this project, as the complexity of DNN, the two DNN include which are DNN sequential model and MLP regressor. The weighted parameters trained automatically by backpropagation and the best DNN would be selected as long as the RMSE is lower.

For DNN sequential model:



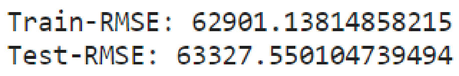
```
Epoch 1/10
794/794 [==============================] - 1s 1ms/step - loss: 684501696.0000 - mean_absolute_error: 13992.518
Epoch 2/10
794/794 [==============================] - 1s 1ms/step - loss: 44405968.0000 - mean_absolute_error: 4589.3315
Epoch 3/10
794/794 [==============================] - 1s 1ms/step - loss: 31747874.0000 - mean_absolute_error: 3833.2930
Epoch 4/10
794/794 [==============================] - 1s 1ms/step - loss: 28106554.0000 - mean_absolute_error: 3626.9644
Epoch 5/10
794/794 [==============================] - 1s 1ms/step - loss: 28555984.0000 - mean_absolute_error: 3653.8342
Epoch 6/10
794/794 [==============================] - 1s 1ms/step - loss: 28160550.0000 - mean_absolute_error: 3609.6641
Epoch 7/10
794/794 [==============================] - 1s 1ms/step - loss: 26966692.0000 - mean_absolute_error: 3527.4973
Epoch 8/10
794/794 [==============================] - 1s 1ms/step - loss: 27232452.0000 - mean_absolute_error: 3543.0874
Epoch 9/10
794/794 [==============================] - 1s 1ms/step - loss: 27537100.0000 - mean_absolute_error: 3569.0974
Epoch 10/10
794/794 [==============================] - 1s 1ms/step - loss: 27463158.0000 - mean_absolute_error: 3560.0803
Train-RMSE: 4915.116991328634
Test-RMSE: 5108.923144118085
```



```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_48 (Dense)            (None, 256)               1280

 dense_49 (Dense)            (None, 256)               65792

 dense_50 (Dense)            (None, 256)               65792

 dense_51 (Dense)            (None, 256)               65792

 dense_52 (Dense)            (None, 256)               65792

 dense_53 (Dense)            (None, 1)                 257

=================================================================
Total params: 264,705
Trainable params: 264,705
Non-trainable params: 0
```

In DNN model, since four features used, there are 4 neurons input layer. 256 neurons in each hidden layers could have much more deep parameters updating for more accurate result and only 1 neuron in output layer is for the only one prediction value of output.

For MLP regressor:



```
Train-RMSE: 62901.13814858215
Test-RMSE: 63327.550104739494
```

Clearly, the DNN sequential model outperforms and its RMSE is largely smaller than the MLP regressor of RMSE. According to the sklearn documentary, the MLP does not have activation in output layer. That’s may become the one of the reasons getting high RMSE in MLP model.

## Summary and future work

To sum up, the regression performance are similar among all the models. For selecting the best model, as DNN model obtained the smallest RMSE, the DNN sequential model is the best model to assist AlphaMoney to predict the loan amount as it is more possible to generalize all the regression problem cases. However, all the models would have large room for improvement for future work. The optimization of hyperparameters tuning could be improving the model accuracy itself. This may require the expensive time costs or computation costs. Moreover, the internal structure of algorithm is still available for doing scaling such as some kernel functions. As this project doesn’t involve these topics, there should have some ways to improve the models even better.

## Reference

AL-Ma'amari, M. (2018, October 25). *Deep neural networks for regression problems*. Medium. https://towardsdatascience.com/deep-neural-networks-for-regression-problems-81321897ca33

Bakshi, C. (2020, June 9). *Random forest regression*. Medium. https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

Chen, T., & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

Girgin, S. (2019, April 5). *Decision tree regression in 6 steps with Python*. Medium. https://medium.com/@sametgirgin/decision-tree-regression-in-6-steps-with-python-c1564b153b74

Krishna, D. (2020, December 23). *A look at the maths behind linear classification*. Medium. https://towardsdatascience.com/a-look-at-the-maths-behind-linear-classification-166e99a9e5fb

Polzer, D. (2022, February 8). *7 of the most used regression algorithms and how to choose the right one*. Medium. https://towardsdatascience.com/7-of-the-most-commonly-used-regression-algorithms-and-how-to-choose-the-right-one-fc3c8890f9e3#7f1c

Raj, A. (2020, October 5). *Unlocking the true power of support vector regression*. Medium. https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0#:~:text=Support%20Vector%20Regression%20is%20a,the%20maximum%20number%20of%20points

Tanaya, P. (2021, June 25). *Linear regression using neural network | Neural network for regression*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/linear-regression-using-neural-networks/