Northeastern University

Report for Assignment_3

Ten uses cases and motivations for building a machine learning model using the Lending club dataset

Team Member: Jixiao Yang

Cuicui Feng

Miner Yang

Yuning Song

Abstract

With the release of Lending club data, a lot of researchers and practitioners are leveraging this dataset to understand credit risk, borrowing patterns, investment opportunities and to understand consumer choice and behaviors. This dataset is a gold mine to teach data science and a lot of schools are using it to teach data science. We scraped a database from Lending club data. And our role is an economic consulting company. Our aim is to recommend the company the best strategy to loan money to the people that can obtain maximum profits. In this assignments we have done:

1. Understanding the client and gather data from Lending club data
2. Data cleansing and data pre-processing. Then use feature engineering to deal with data. Prepare the difference between the two methods.
3. Use 3 models (Regression, Random forest and Neural Network) to predict our loan strategy among people in different credit grades.
4. Use hyper-parameter tuning and the methods of AutoML to optimize the above three models.
5. Analyze the all above data , draw some diagrams about that and then build some test to ensure that the output is important to the client.

<u>Task 1: Framing</u>

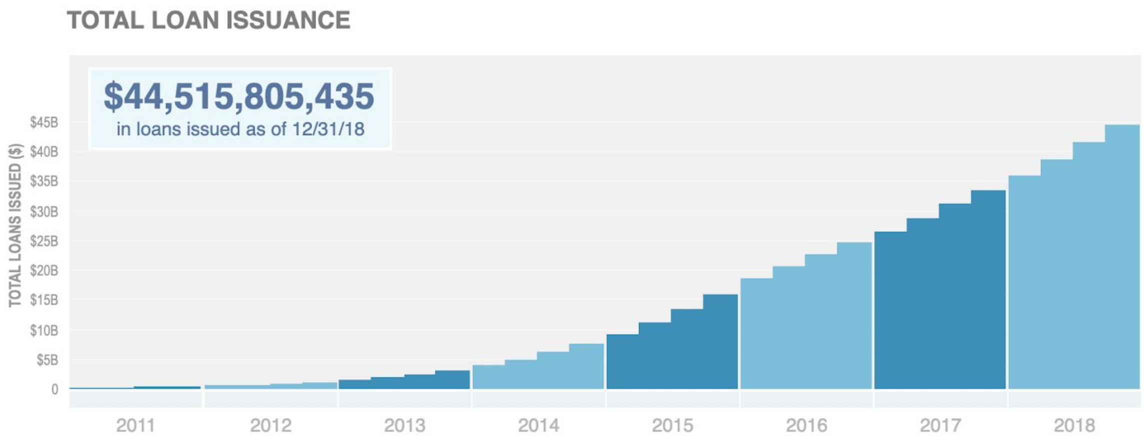
PART 1: Research and summarize what are the client's needs.

Our client is an economic consulting firm which interested in understanding how customers of lending club can provide leading indicator on the economy and lending. In order to understand our client, we make an explanation about what is main economic consulting. Economic consulting is a type of paralegal work, it offers expert to testify in litigation matters related to economics and finance. In practical terms, it means economic consulting firms produce research, analysis, and testimony to be used in court cases.

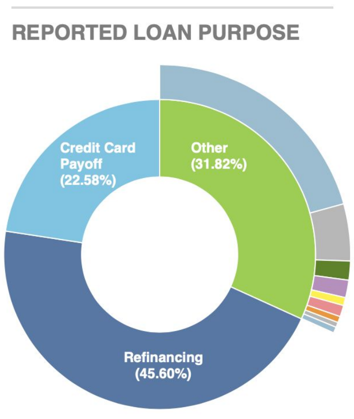
Take SMITH Economics Group as an example to analyze the client's needs. SMITH Group deliver top-notch economic and financial analysis, testimony, problem solving and creativity. Now, it wants to obtain leading indicator on the economy and lending, and it chooses to complete this case study according to the dataset of customers of Lending club. Therefore, our goal is to find the most stable customer's attributes at the same time they can get more profits as possible. At last, we feedback our model about predict rates and teach client how to use the model.

PART 2: Conclude keys from the graphs which Lending club has generated.

At first, we summarize the three charts about Lending club's overall situation.

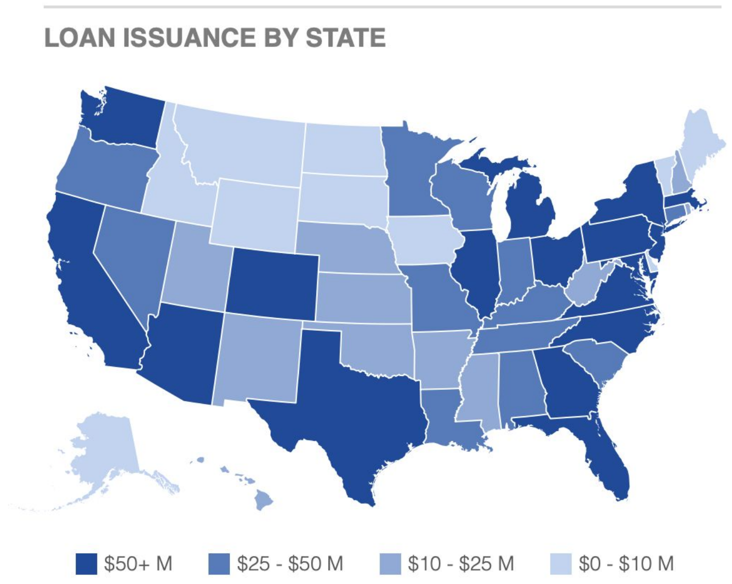


The overall scale of the platform's main point is the total number of loans per year. According to TOTAL LOAN INSURANCE given by the platform, we can know that the total loan issuance is gradually increasing every year, and the rate of increase is accelerating. It represents that the company still has a lot of space to improvement and the market is still not saturated.



68.18% of LendingClub borrowers report using their loans to refinance existing loans or pay off their credit cards as of 12/31/18.[1]

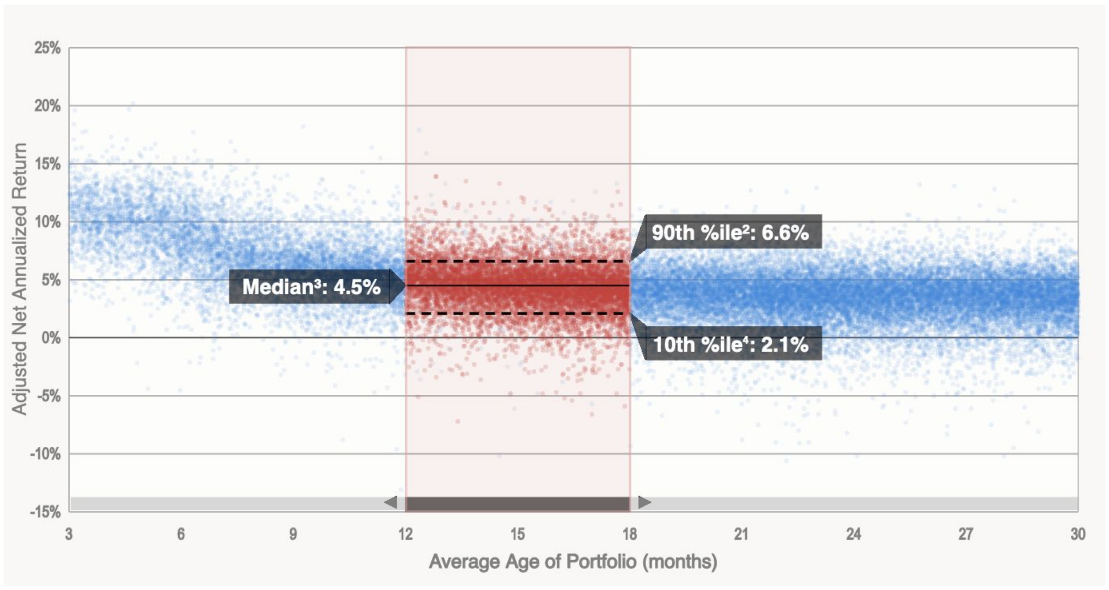
According to the lender's main purpose, we can know 63%, more than half of them are for refinancing existing loans or pay off their credit cards.



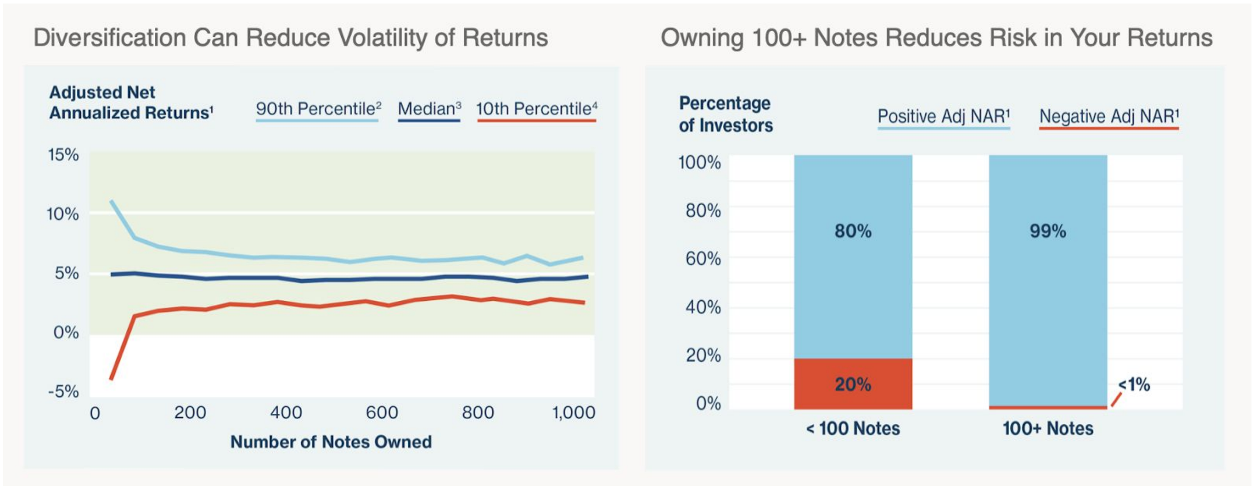
Most recent reported quarter ending 12/31/18

Depending on the total amount of loans in different states, we can briefly judge that the degree of development is probably related to the economic level of each state. But if we want to analyze it further, we need to calculate the parameters such as the population density of each state.

Second, there are two factors influence the Adjusted Net Annualized Return, the average age of portfolio and diversification of investment.

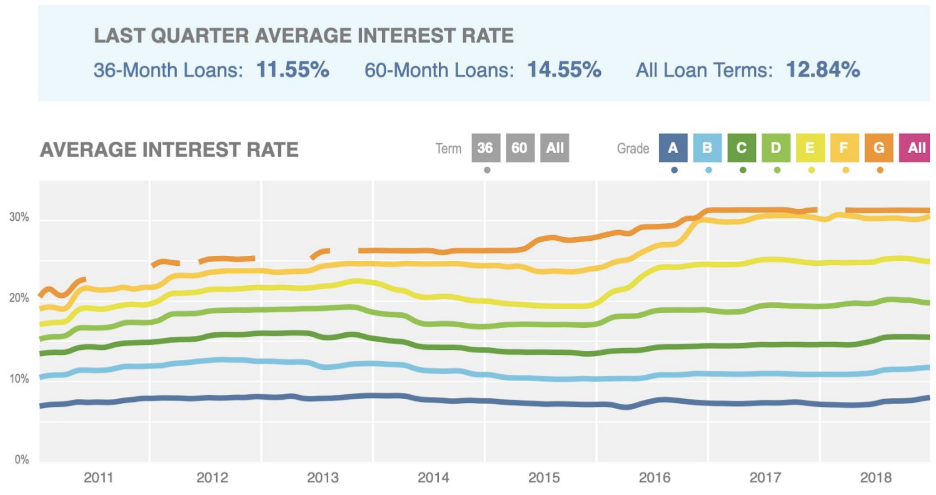


According to the Average Age of Portfolio, we can know that the higher average age of the Notes in the portfolio, the more stable the annual interest rate of the investment. Basically, after nine months, you can get a return approximately 5 percent. On the one hand, we can simply summarize that investors have determined their investment methods based on previous investment experience, and in order to maintain stability, their interest rates have also stabilized.

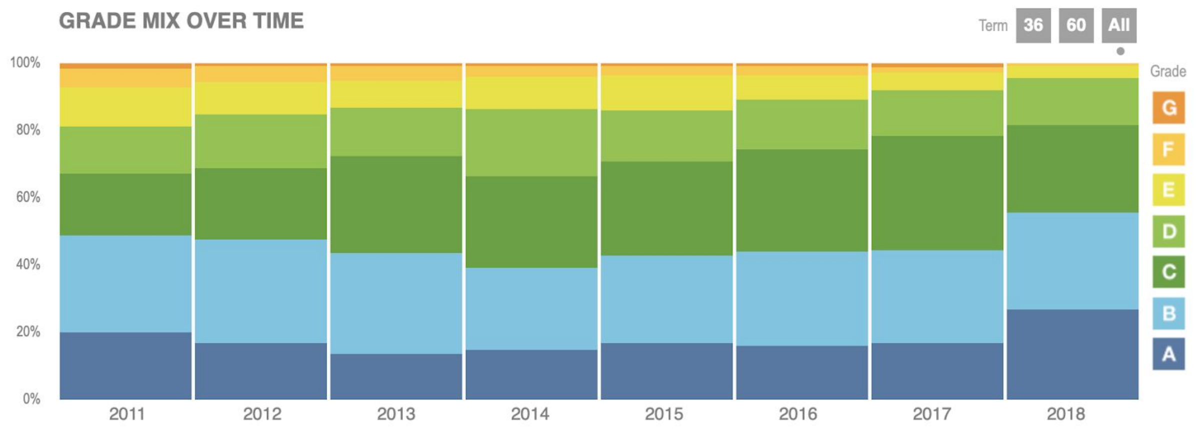


In addition to the investment time, another aspect we need to pay attention to is the diversification of investment. Diversified accounts have generally experienced less volatility and more solid returns than investors with more concentrated holdings. So, for risk-averse investors, it is a good idea to choose to invest their own funds separately.

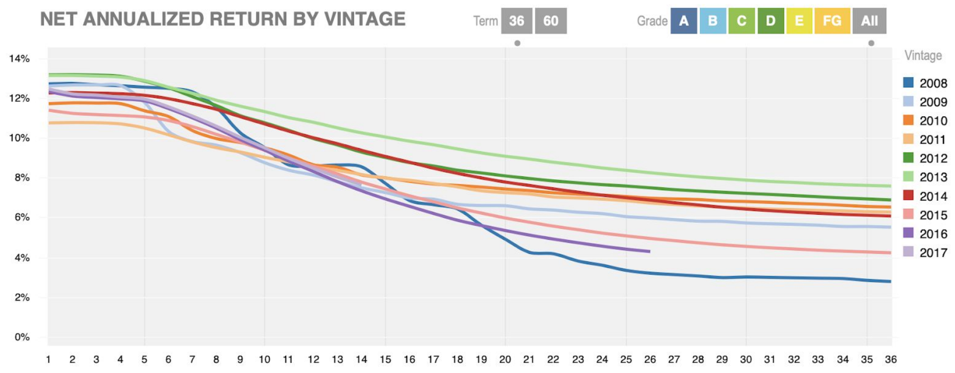
Third, there are four charts to describe different grade's development and composition.



The average annual interest rate for different grades has always had a clear dividing line. Now the interest rate between different grades difference 4%. B is higher 4% than A and C is higher than 4% more than B.



The composition of different levels also has significant changes. There are more and more people choosing ABC grades with high stability. High-risk levels are gradually abandoned by the public.



According to the NET ANNUALIZED RETURN BY VINTAGE, returns are gradually decreasing every year.

PART 3: Explore the data and comment on the data quality and features.

All the data we have including 890 thousand records formed by 75 attributes. The data dictionary provides ample explanation for the meanings of each attribute. We need to understand the data dictionary at first, and then manual analysis the dataset.

At first, in order to complete data cleaning, we should know that real world data tend to be incomplete, noisy and inconsistent. Data cleaning routines attempt to fill in missing value, smooth out noise while identifying outliers, and correct inconsistent in the data. There are 78 variables in the data dictionary, and with 4 meaningless which means there are no corresponding attributes in the table. And then we can find there are 20 variables are mainly empty so we can delete it first. There are 55 remaining attributes to analysis.

At second, we classify dataset variables according to the actual meaning of the attributes and delete meaningless attributes. We divided the remained 21 attributes into three classes.

- ★ The data related to the member's basic information and inherent capital.
- ★ The data related to the grade and interest rate.
- ★ The data related to the existing lending record's state.

Task 2: Data preparation

PART 1: Data cleaning, pre-processing and do feature engineering

There are 890,000 records with 75 variables, so we need spend great number of times to process the data cleaning.

At first, we delete the records that more than 50% are empty.

```sql
select count(*) from info6105.loan where mths_since_last_record is null;        -- 750330 null
select count(*) from info6105.loan where `desc` is null;                        -- 761354 null
select count(*) from info6105.loan where
open_acc_6m is null or
open_il_6m is null or
open_il_12m is null or
open_il_24m is null or
total_bal_il is null or
open_rv_12m is null or
open_rv_24m is null or
max_bal_bc is null or
all_util is null or
inq_fi is null or
total_cu_tl is null or
inq_last_12m is null;                                                           -- 866011 null
select count(*) from info6105.loan where mths_since_rcnt_il is null;            -- 866573 null
select count(*) from info6105.loan where verification_status_joint is null;     -- 886872 null
```

Second, we deal with the data is not directly related to the this analysitic.

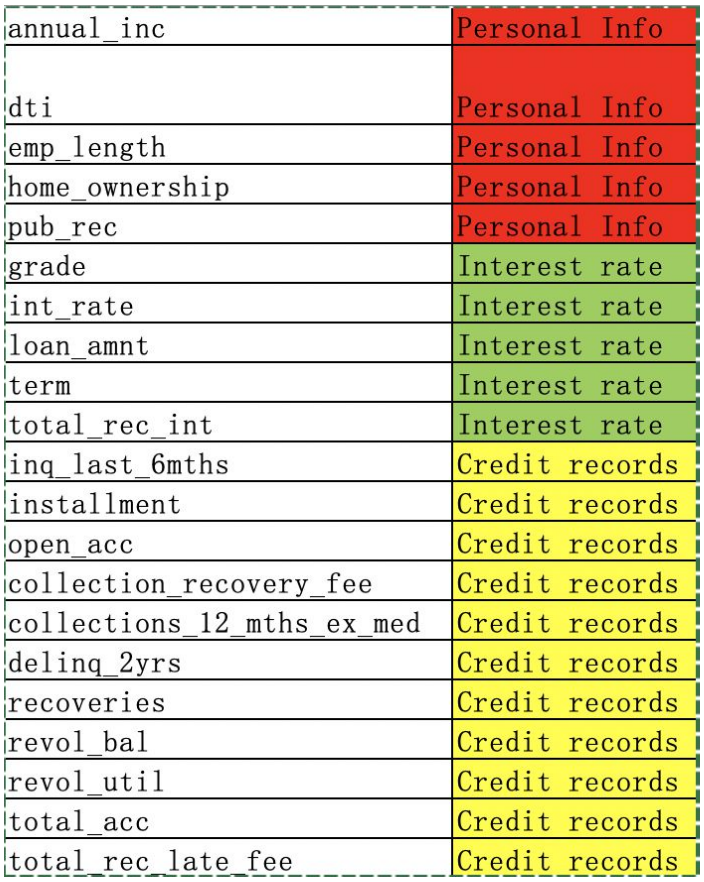
next_pymnt_d	Next scheduled payment date
open_acc	The number of open credit lines in the borrower's credit file.

total_acc	The total number of credit lines currently in the borrower's credit file
total_pymnt	Payments received to date for total amount funded

Third, there are individual and joint situations, we mainly analysis about the individual transaction.

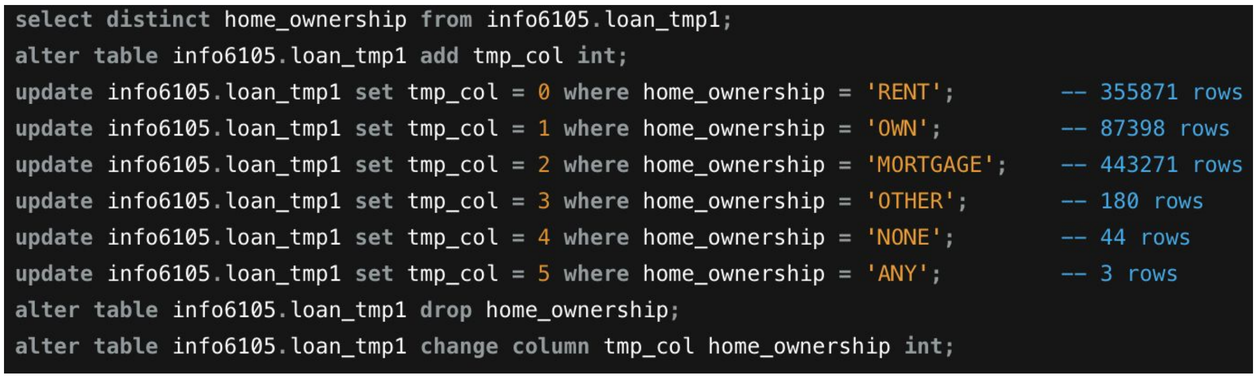
open_acc_6m	Number of open trades in last 6 months
open_il_6m	Number of currently active installment trades
open_il_12m	Number of installment accounts opened in past 12 months
open_il_24m	Number of installment accounts opened in past 24 months
mths_since_rcnt_il	Months since most recent installment accounts opened
total_bal_il	Total current balance of all installment accounts
il_util	Ratio of total current balance to high credit/credit limit on all install acct
open_rv_12m	Number of revolving trades opened in past 12 months
open_rv_24m	Number of revolving trades opened in past 24 months
max_bal_bc	Maximum current balance owed on all revolving accounts
all_util	Balance to credit limit on all trades
total_rev_hi_lim	Total revolving high credit/credit limit
inq_fi	Number of personal finance inquiries
total_cu_tl	Number of finance trades
inq_last_12m	Number of credit inquiries in past 12 months

Fourth, we conclude some key attributes to continue next steps.



annual_inc	Personal Info
dti	Personal Info
emp_length	Personal Info
home_ownership	Personal Info
pub_rec	Personal Info
grade	Interest rate
int_rate	Interest rate
loan_amnt	Interest rate
term	Interest rate
total_rec_int	Interest rate
inq_last_6mths	Credit records
installment	Credit records
open_acc	Credit records
collection_recovery_fee	Credit records
collections_12_mths_ex_med	Credit records
delinq_2yrs	Credit records
recoveries	Credit records
revol_bal	Credit records
revol_util	Credit records
total_acc	Credit records
total_rec_late_fee	Credit records

At last, because some of attributes are String or some other types, we need to transform them to integer state. In order to prepare for the data modeling.



```
select distinct home_ownership from info6105.loan_tmp1;
alter table info6105.loan_tmp1 add tmp_col int;
update info6105.loan_tmp1 set tmp_col = 0 where home_ownership = 'RENT';        -- 355871 rows
update info6105.loan_tmp1 set tmp_col = 1 where home_ownership = 'OWN';         -- 87398 rows
update info6105.loan_tmp1 set tmp_col = 2 where home_ownership = 'MORTGAGE';    -- 443271 rows
update info6105.loan_tmp1 set tmp_col = 3 where home_ownership = 'OTHER';       -- 180 rows
update info6105.loan_tmp1 set tmp_col = 4 where home_ownership = 'NONE';        -- 44 rows
update info6105.loan_tmp1 set tmp_col = 5 where home_ownership = 'ANY';         -- 3 rows
alter table info6105.loan_tmp1 drop home_ownership;
alter table info6105.loan_tmp1 change column tmp_col home_ownership int;
```

PART 2: Use Featuretools process data preparation.

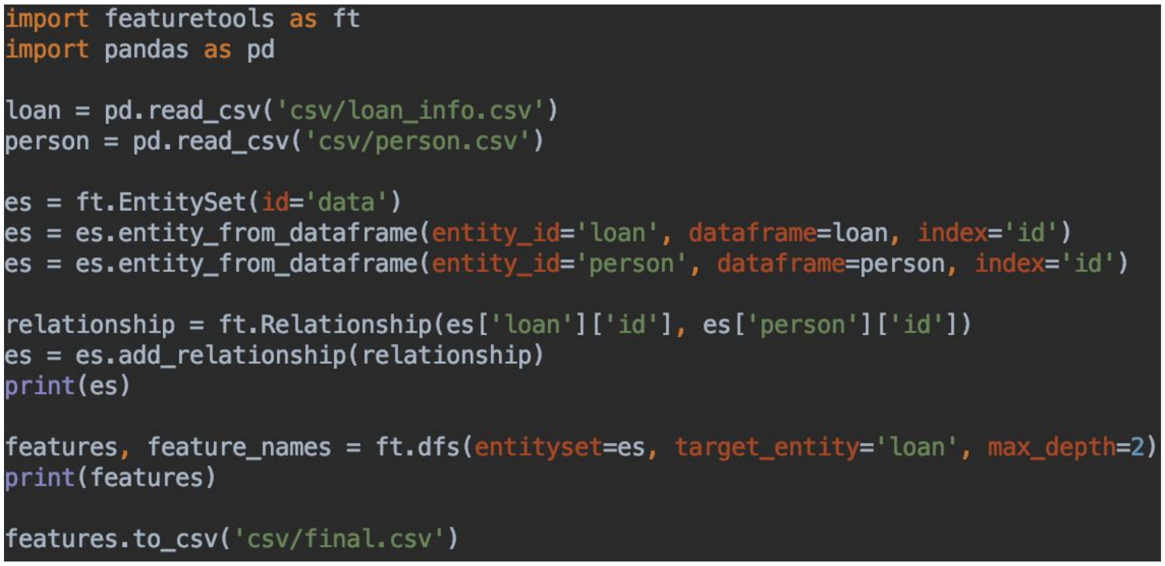
What is Featuretools?

Featuretools is a framework to perform automated feature engineering. It excels at transforming temporal and relational datasets into feature matrices for machine learning.

Install Featuretools.

python -m pip install featuretools

Specific implementations.



```python
import featuretools as ft
import pandas as pd

loan = pd.read_csv('csv/loan_info.csv')
person = pd.read_csv('csv/person.csv')

es = ft.EntitySet(id='data')
es = es.entity_from_dataframe(entity_id='loan', dataframe=loan, index='id')
es = es.entity_from_dataframe(entity_id='person', dataframe=person, index='id')

relationship = ft.Relationship(es['loan']['id'], es['person']['id'])
es = es.add_relationship(relationship)
print(es)

features, feature_names = ft.dfs(entityset=es, target_entity='loan', max_depth=2)
print(features)

features.to_csv('csv/final.csv')
```

We split the raw data into two parts: personal information and loan information. Then according to the featuretools, we can obtain an feature matrix which contains the features we need.

PART 3: Note on our observations on featureTools vs manual Feature Engineering

Feature engineering is a process of collecting data sets and constructing explanatory variables (features) that can be used to train machine learning models (for predicting problems).

The traditional feature engineering method is based on the single-time construction of domain knowledge, which is manual feature engineering, which is cumbersome, time-consuming and error-prone. The code for manual feature engineering is related to a specific problem, and you need to write different code for different data sets.

Automated feature engineering uses frameworks to automatically extract useful and meaningful features from a set of related data tables, improving traditional standard workflows, and frameworks can be used to solve different problems. Not only does it save time, it also creates interpretable features and prevents data leakage by filtering time-related data.

Task 3: Prediction

Basically, we have some features that matters to our predicted target feature. In our case, as a economic consulting company in this lending club, we need to evaluate the lending feasibility grade of customers(the borrower) based on their information. So in the trained model ,if we want to predict borrowers' grade, then quantified them and put these numbers into our trained model, as we all know we have 3 manual model and 3 antoML then the predict method will give a distinct integer from 0 to 6 as our final feasibility grade for loan to refer. the lower the grade is , the lending feasibility it will be.
For the interest rate prediction, we want to build three kinds of models and compare their performances.

Model design

According to the features we selected in Task 2, we choose the independent and dependent variables for the input of the models as below:

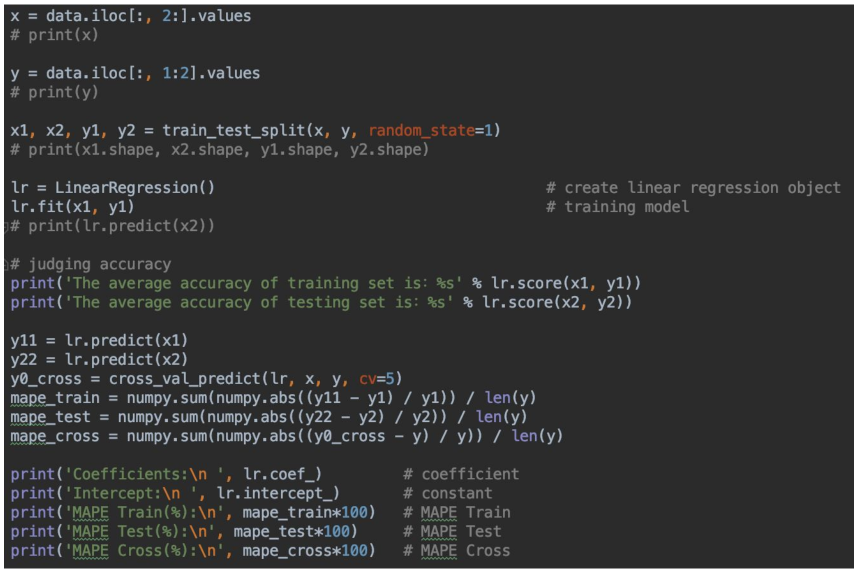
- Dependent variable: int_rate
- Independent variables: other fields (grade, loan_amnt, term, annual_inc...)

So, we want to use the machine learning models to figure out the relationship between interest rate and other variables, so that after building, training and testing the models, we can predict the interest rate of loans.

Model 1: Linear Regression

One of the simplest and most popular modeling methods is linear regression. Linear regression fits a straight line (known linear function) to a set of data values. The form of the function fitted by linear regression is:

y = a0 + a1*x1 + a2*x2 + …

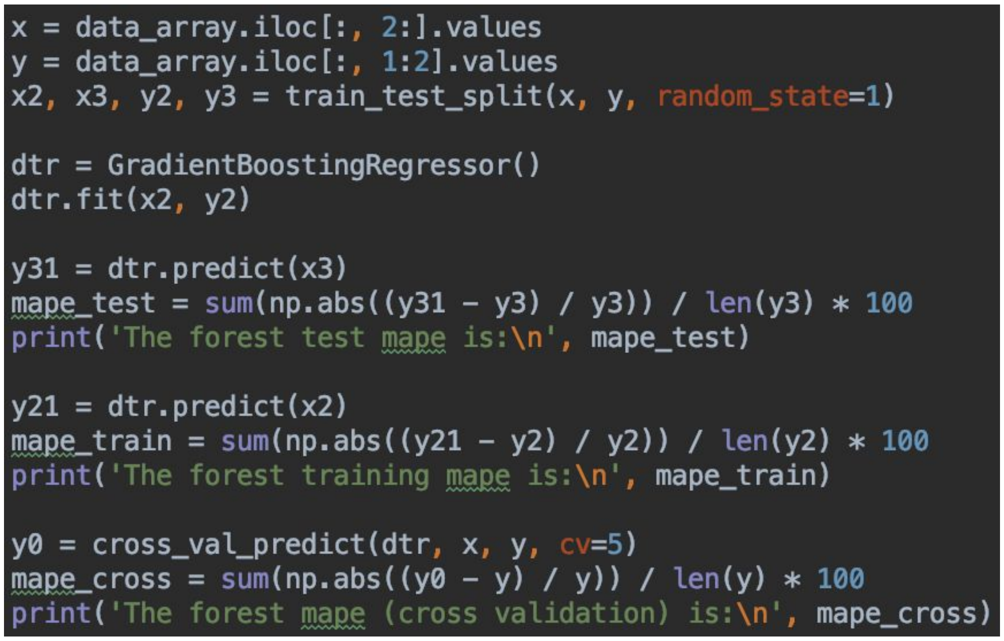


```
x = data.iloc[:, 2:].values
# print(x)

y = data.iloc[:, 1:2].values
# print(y)

x1, x2, y1, y2 = train_test_split(x, y, random_state=1)
# print(x1.shape, x2.shape, y1.shape, y2.shape)

lr = LinearRegression()                              # create linear regression object
lr.fit(x1, y1)                                       # training model
# print(lr.predict(x2))

# judging accuracy
print('The average accuracy of training set is: %s' % lr.score(x1, y1))
print('The average accuracy of testing set is: %s' % lr.score(x2, y2))

y11 = lr.predict(x1)
y22 = lr.predict(x2)
y0_cross = cross_val_predict(lr, x, y, cv=5)
mape_train = numpy.sum(numpy.abs((y11 - y1) / y1)) / len(y)
mape_test = numpy.sum(numpy.abs((y22 - y2) / y2)) / len(y)
mape_cross = numpy.sum(numpy.abs((y0_cross - y) / y)) / len(y)

print('Coefficients:\n ', lr.coef_)         # coefficient
print('Intercept:\n ', lr.intercept_)       # constant
print('MAPE Train(%):\n', mape_train*100)   # MAPE Train
print('MAPE Test(%):\n', mape_test*100)     # MAPE Test
print('MAPE Cross(%):\n', mape_cross*100)   # MAPE Cross
```

testing_data	9.7215781
training _data	9.74099664
after_5-fold cross validation	9.90996087

Model 2: Random Forest

The random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:
g(x)=f0(x)+f1(x)+f2(x)+...
where the final model g is the sum of simple base models fi. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called model ensembling. In random forests, all the base models are constructed independently using a different subsample of the data.

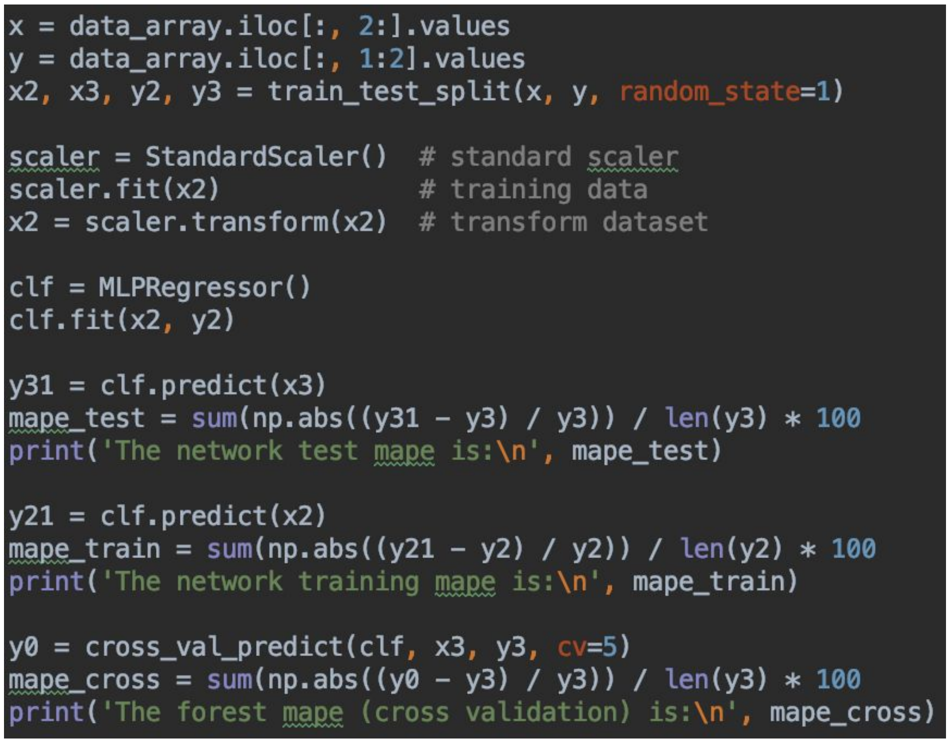


```
x = data_array.iloc[:, 2:].values
y = data_array.iloc[:, 1:2].values
x2, x3, y2, y3 = train_test_split(x, y, random_state=1)

dtr = GradientBoostingRegressor()
dtr.fit(x2, y2)

y31 = dtr.predict(x3)
mape_test = sum(np.abs((y31 - y3) / y3)) / len(y3) * 100
print('The forest test mape is:\n', mape_test)

y21 = dtr.predict(x2)
mape_train = sum(np.abs((y21 - y2) / y2)) / len(y2) * 100
print('The forest training mape is:\n', mape_train)

y0 = cross_val_predict(dtr, x, y, cv=5)
mape_cross = sum(np.abs((y0 - y) / y)) / len(y) * 100
print('The forest mape (cross validation) is:\n', mape_cross)
```

testing_data	8.593283106658427
training _data	8.624867887487639
after_5-fold cross validation	9.605741596554422

Model 3: Neural networks

Neural networks (also called “multilayered perceptron”) provide models of data relationships through highly interconnected, simulated “neurons” that accept inputs, apply weighting coefficients and feed their output to other “neurons” which continue the process through the network to the eventual output. Some neurons may send feedback to earlier neurons in the network. Neural networks are “trained” to deliver the desired result by an iterative (and often lengthy) process where the weights applied to each input at each neuron are adjusted to optimize the desired output.



```
x = data_array.iloc[:, 2:].values
y = data_array.iloc[:, 1:2].values
x2, x3, y2, y3 = train_test_split(x, y, random_state=1)

scaler = StandardScaler()  # standard scaler
scaler.fit(x2)             # training data
x2 = scaler.transform(x2)  # transform dataset

clf = MLPRegressor()
clf.fit(x2, y2)

y31 = clf.predict(x3)
mape_test = sum(np.abs((y31 - y3) / y3)) / len(y3) * 100
print('The network test mape is:\n', mape_test)

y21 = clf.predict(x2)
mape_train = sum(np.abs((y21 - y2) / y2)) / len(y2) * 100
print('The network training mape is:\n', mape_train)

y0 = cross_val_predict(clf, x3, y3, cv=5)
mape_cross = sum(np.abs((y0 - y3) / y3)) / len(y3) * 100
print('The forest mape (cross validation) is:\n', mape_cross)
```

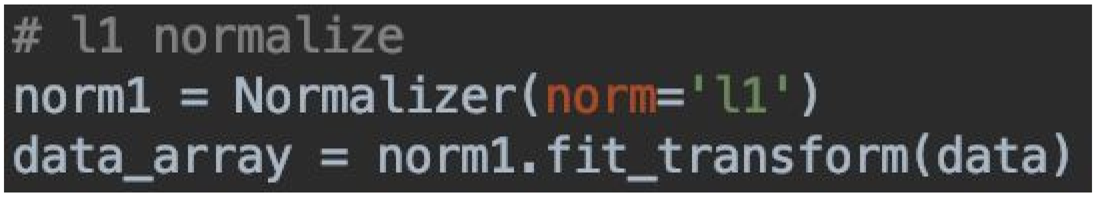
testing_data	7.4511299212
training _data	7.3820697173032965
after_5-fold cross validation	8.010593620001984

Task 4:Hyper-parameter optimization

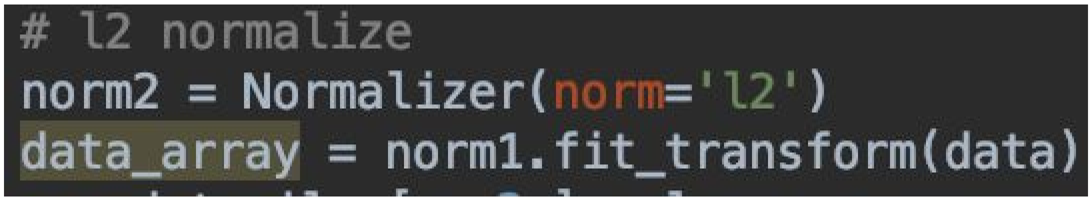
PART 1: Hyperparameter tuning

Model training is a process that minimizes the loss function (or cost function, evaluation index of the training phase). This process uses many optimization techniques and methods, and some parameters are needed in the optimization method used.

Method 1: Regression



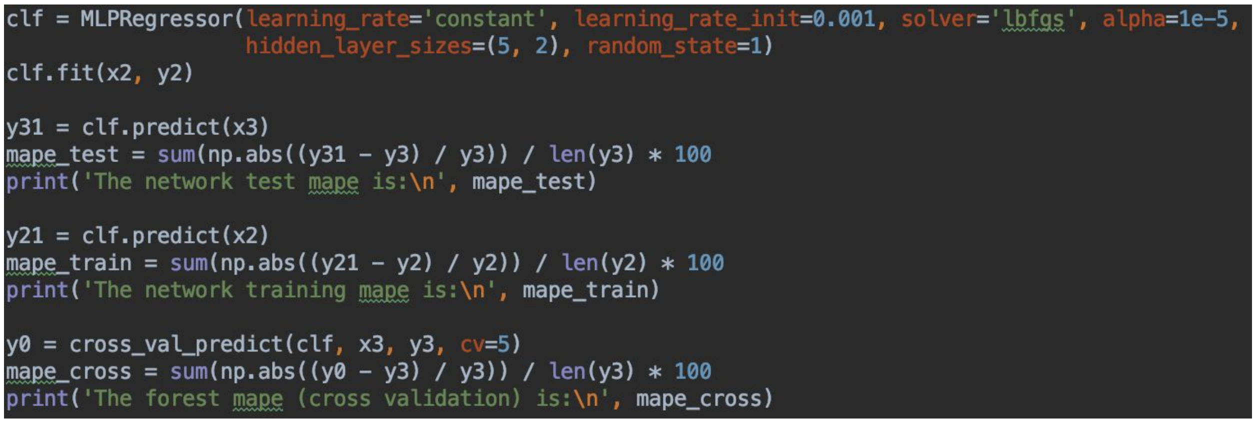
```
# l1 normalize
norm1 = Normalizer(norm='l1')
data_array = norm1.fit_transform(data)
```



```
# l2 normalize
norm2 = Normalizer(norm='l2')
data_array = norm1.fit_transform(data)
```

	L1 Regularization	L2 Regularization
testing_data	9.39761505	9.51184836
training _data	9.40653438	9.51543798
after_5-fold cross validation	9.51198772	9.65471118

Method 2: Neural Networks

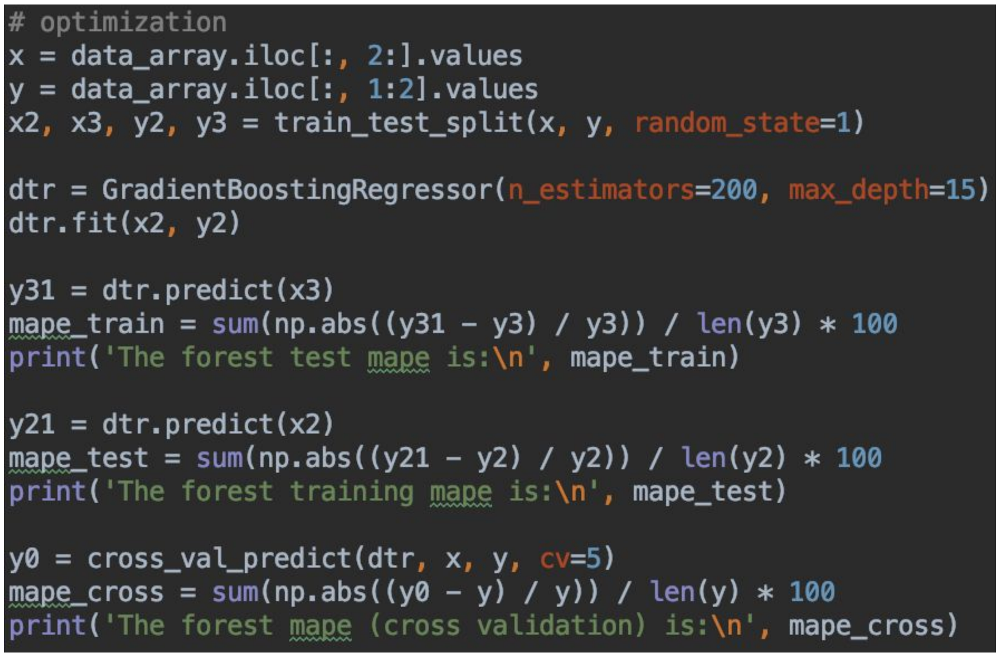


```
clf = MLPRegressor(learning_rate='constant', learning_rate_init=0.001, solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(5, 2), random_state=1)
clf.fit(x2, y2)

y31 = clf.predict(x3)
mape_test = sum(np.abs((y31 - y3) / y3)) / len(y3) * 100
print('The network test mape is:\n', mape_test)

y21 = clf.predict(x2)
mape_train = sum(np.abs((y21 - y2) / y2)) / len(y2) * 100
print('The network training mape is:\n', mape_train)

y0 = cross_val_predict(clf, x3, y3, cv=5)
mape_cross = sum(np.abs((y0 - y3) / y3)) / len(y3) * 100
print('The forest mape (cross validation) is:\n', mape_cross)
```

clf = MLPRegressor(learning_rate='constant',learning_rate_init=0.001, solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1)
Learning_rate is set to constant, initial learning_rate is 0.01, learning method：lbfgs, accuracy is 1e-5, hidden_layer_sizes: two layers, first layer 5 nodes second layer 2 nodes

testing_data	7.7160901213
training _data	7.024395577962459
after_5-fold cross validation	8.429980757691716

Method 3: Random Forests



```
# optimization
x = data_array.iloc[:, 2:].values
y = data_array.iloc[:, 1:2].values
x2, x3, y2, y3 = train_test_split(x, y, random_state=1)

dtr = GradientBoostingRegressor(n_estimators=200, max_depth=15)
dtr.fit(x2, y2)

y31 = dtr.predict(x3)
mape_train = sum(np.abs((y31 - y3) / y3)) / len(y3) * 100
print('The forest test mape is:\n', mape_train)

y21 = dtr.predict(x2)
mape_test = sum(np.abs((y21 - y2) / y2)) / len(y2) * 100
print('The forest training mape is:\n', mape_test)

y0 = cross_val_predict(dtr, x, y, cv=5)
mape_cross = sum(np.abs((y0 - y) / y)) / len(y) * 100
print('The forest mape (cross validation) is:\n', mape_cross)
```

There are 200 trees, each with a depth of 15.

testing_data	7.521549984792737
training _data	7.501931441158526
after_5-fold cross validation	8.642514223451344

The effect of Hyper parameter tuning

Hyperparameters are the adjustment knobs that control our model structure, function, efficiency, etc. Such as learning rate, epochs, num of hidden layers, num of hidden layer units.

For hyperparameter optimization, the objective function is the verification error of a machine learning model using a set of hyperparameters. Its goal is to find the hyperparameters that produce the smallest error on the validation set and hope to generalize these results onto the test set. The overhead of evaluating the objective function is huge because it requires training a machine learning model with a specific set of hyperparameters. Ideally, we would like to find a way to explore both the search space and the time-consuming hyperparameter evaluation.

Nowadays more and more hyperparameter tuning processes are done through automated methods, which aim to find optimal hyperparameters in a shorter time using heuristic search with strategy, in addition to the initial settings, and No additional manual operation is required.

PART 2: AutoML Methods.

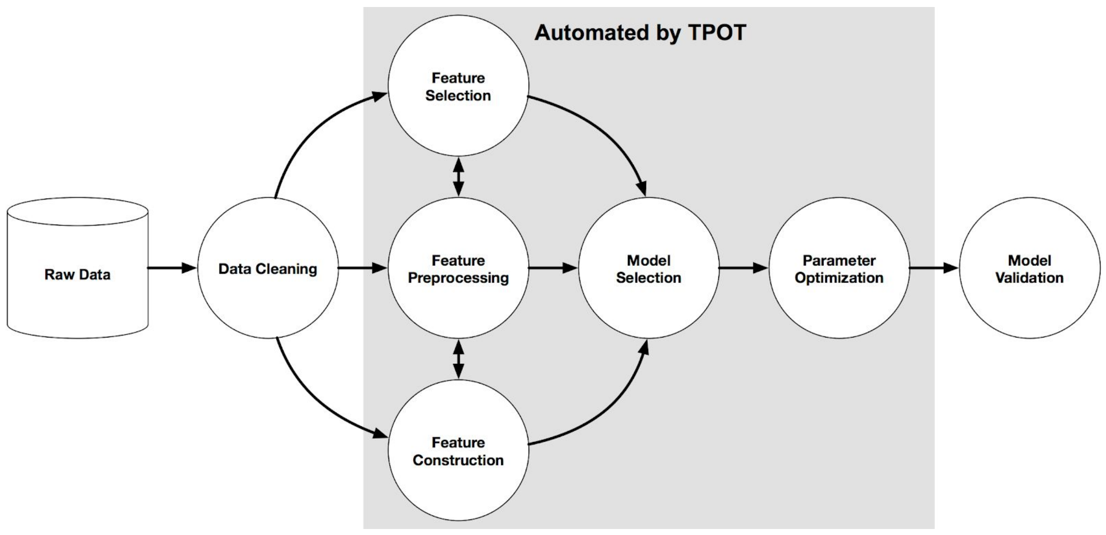
Automated Machine Learning provides methods and processes to make Machine Learning available for non-Machine Learning experts, to improve efficiency of Machine Learning and to accelerate research on Machine Learning.

- ★ **Method 1: TPOT**

What is TPOT?

TPOT is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming.

What is the main role of the TPOT?



What is the benefit of the TPOT?

TPOT will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for your data.

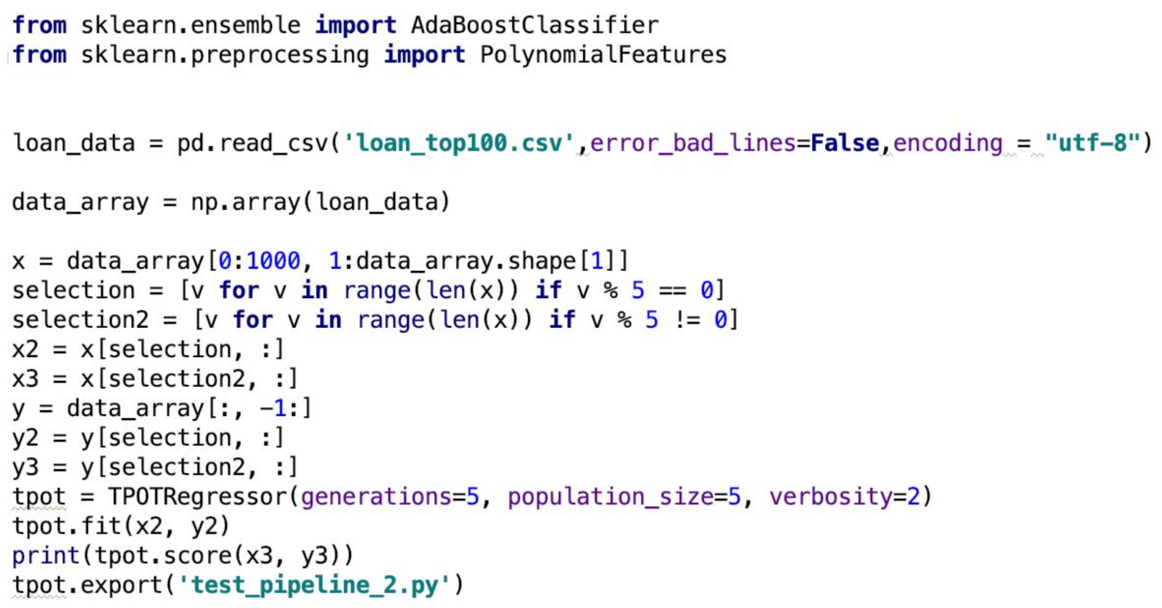
The step of install TPOT.

Step 1 : Install Python Library.

I. pip install NumPy
II. pip install SciPy
III. pip install Scikit-learn
IV. pip install DEAP
V. pip install update_checker
VI. pip install tqdm

Step 2 : Install TPOT.

pip install TPOT

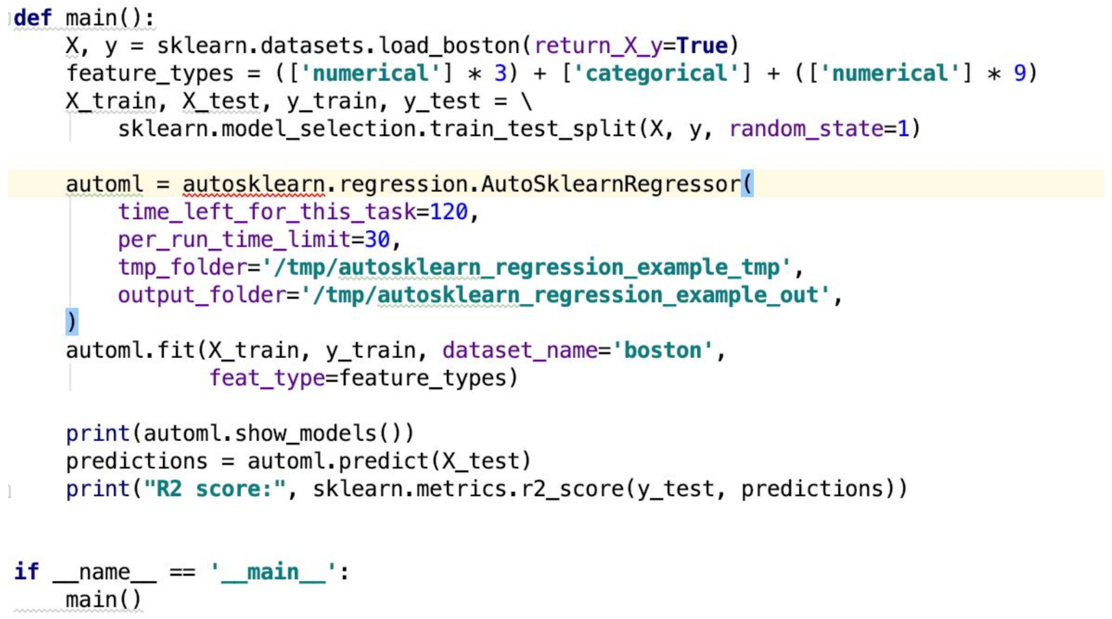


```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.preprocessing import PolynomialFeatures


loan_data = pd.read_csv('loan_top100.csv',error_bad_lines=False,encoding = "utf-8")

data_array = np.array(loan_data)

x = data_array[0:1000, 1:data_array.shape[1]]
selection = [v for v in range(len(x)) if v % 5 == 0]
selection2 = [v for v in range(len(x)) if v % 5 != 0]
x2 = x[selection, :]
x3 = x[selection2, :]
y = data_array[:, -1:]
y2 = y[selection, :]
y3 = y[selection2, :]
tpot = TPOTRegressor(generations=5, population_size=5, verbosity=2)
tpot.fit(x2, y2)
print(tpot.score(x3, y3))
tpot.export('test_pipeline_2.py')
```

★ **Method 2: AutSKLearn**

What is AutSKLearn?

Auto-sklearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator



```python
def main():
    X, y = sklearn.datasets.load_boston(return_X_y=True)
    feature_types = (['numerical'] * 3) + ['categorical'] + (['numerical'] * 9)
    X_train, X_test, y_train, y_test = \
        sklearn.model_selection.train_test_split(X, y, random_state=1)

    automl = autosklearn.regression.AutoSklearnRegressor(
        time_left_for_this_task=120,
        per_run_time_limit=30,
        tmp_folder='/tmp/autosklearn_regression_example_tmp',
        output_folder='/tmp/autosklearn_regression_example_out',
    )
    automl.fit(X_train, y_train, dataset_name='boston',
               feat_type=feature_types)

    print(automl.show_models())
    predictions = automl.predict(X_test)
    print("R2 score:", sklearn.metrics.r2_score(y_test, predictions))


if __name__ == '__main__':
    main()
```

★ **Method 3: H2o.ai**

What is H20.ai?

H2O is a fully open source, distributed in-memory machine learning library with linear scalability. H2O supports the most widely used statistical & machine learning algorithms including gradient boosted machines, generalized linear models, deep learning and more. H2O also has an industry leading AutoML functionality that automatically runs through all the algorithms and their hyperparameters to produce a leaderboard of the best models.

H2O's AutoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time limit. Stacked Ensembles will be automatically trained on collections of individual models to produce highly predictive ensemble models which, in most cases, will be the top performing models in the AutoML Leaderboard.

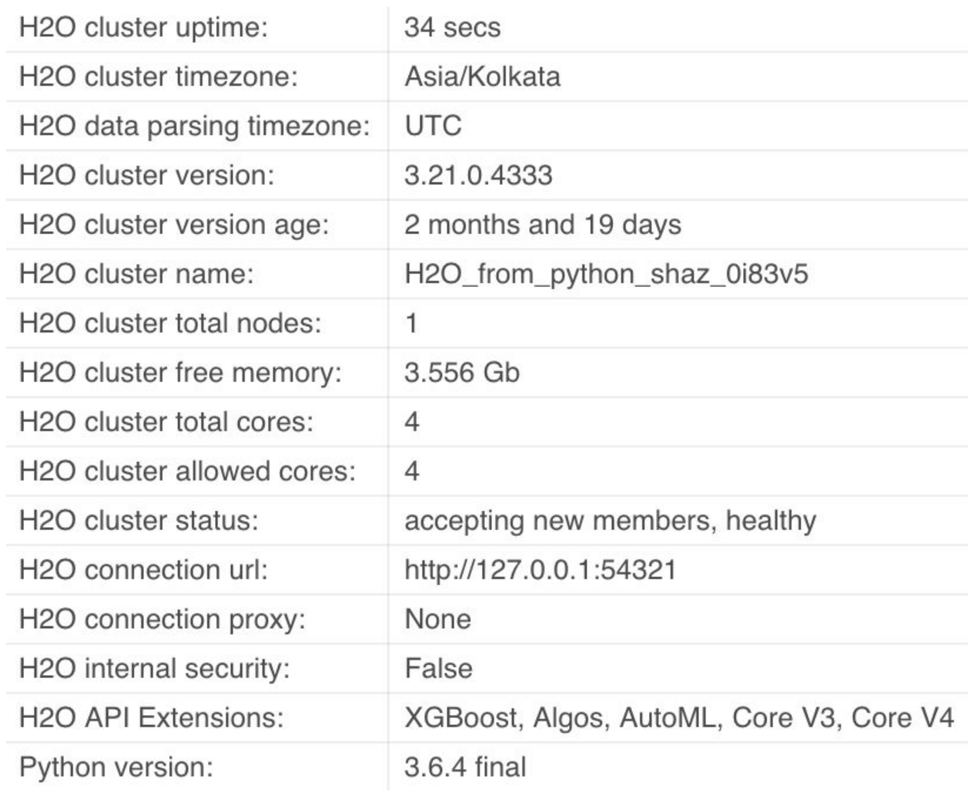
The step of install H2o.ai

```
pip install requests
pip install tabulate
pip install "colorama>=0.3.8"
pip install future

pip install -f http://h2o-
release.s3.amazonaws.com/h2o/latest_stable_Py.html h2o
```

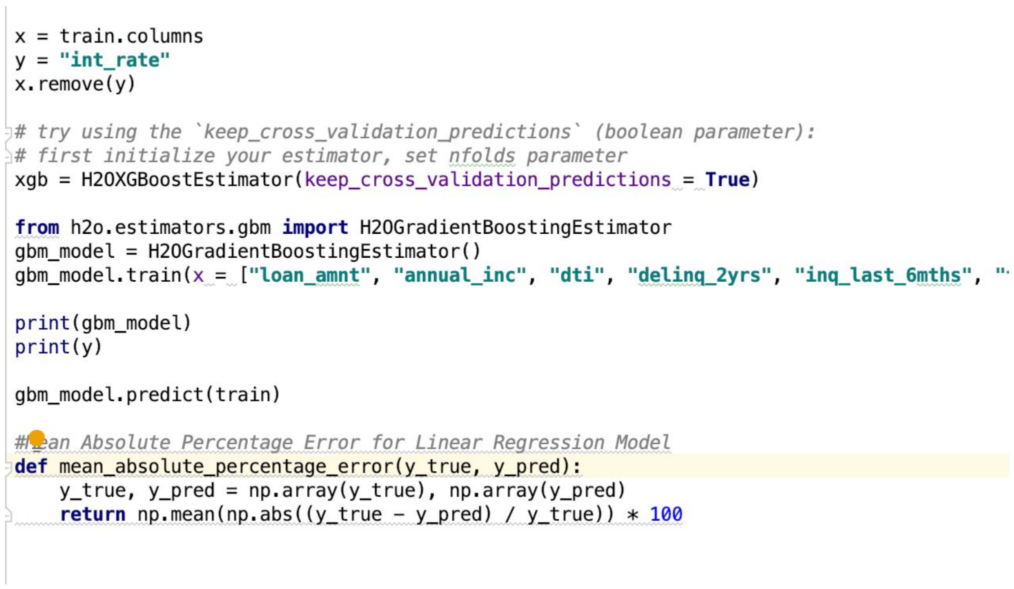
If the setup was successful then will see the following cluster information.

The interface of H2o.ai

The H2O AutoML interface is designed to have as few parameters as possible so that all the user needs to do is point to their dataset, identify the response column and optionally specify a time constraint or limit on the number of total models trained.In both the R and Python API, AutoML uses the same data-related arguments, x, y, training_frame, validation_frame, as the other H2O algorithms. Most of the time, all you'll need to do is specify the data arguments. You can then configure values for max_runtime_secs and/or max_models to set explicit time or number-of-model limits on your run.



H2O cluster uptime:	34 secs
H2O cluster timezone:	Asia/Kolkata
H2O data parsing timezone:	UTC
H2O cluster version:	3.21.0.4333
H2O cluster version age:	2 months and 19 days
H2O cluster name:	H2O_from_python_shaz_0i83v5
H2O cluster total nodes:	1
H2O cluster free memory:	3.556 Gb
H2O cluster total cores:	4
H2O cluster allowed cores:	4
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:54321
H2O connection proxy:	None
H2O internal security:	False
H2O API Extensions:	XGBoost, Algos, AutoML, Core V3, Core V4
Python version:	3.6.4 final



```
x = train.columns
y = "int_rate"
x.remove(y)

# try using the `keep_cross_validation_predictions` (boolean parameter):
# first initialize your estimator, set nfolds parameter
xgb = H2OXGBoostEstimator(keep_cross_validation_predictions = True)

from h2o.estimators.gbm import H2OGradientBoostingEstimator
gbm_model = H2OGradientBoostingEstimator()
gbm_model.train(x = ["loan_amnt", "annual_inc", "dti", "delinq_2yrs", "inq_last_6mths", "

print(gbm_model)
print(y)

gbm_model.predict(train)

#Mean Absolute Percentage Error for Linear Regression Model
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

The interpretability of H2o.ai

In general, linear models focus on understanding and predicting average behavior, whereas machine-learned response functions can often make accurate, but more difficult to explain, predictions for subtler aspects of modeled phenomenon. In a sense, linear models are approximate but create very exact explanations, whereas machine learning can train more exact models but enables only approximate explanations.

The criteria to guarantee reproducibility of H2o.ai

- Same training data. Note: If you have H2O import a whole directory with multiple files instead of a single file, we do not guarantee reproducibility because the data may be shuffled during import.
- Same parameters are used to train the model.
- Same seed is used if sampling is done.

MAPE of three methods:

AutoML Methods	MAPE (%)
TPOT	0.78
AutoSKLearn	0.86
H2o.ai	0.54

PART 3: Discussion

Interpretability

Basically, we have some features that matters to our predicted target feature. In our case, as a economic consulting company in this lending club, we need to evaluate the lending feasibility grade of customers(the borrower) based on their information. So in the trained model ,if we want to predict borrowers' grade, we need features like …..,then quantified them and put these numbers into our trained model, as we all know we have 3 manual model and 3 antoML

then the predict method will give a distinct integer from 0 to 6 as our final feasibility grade for

loan to refer. the lower the grade is , the lending feasibility it will be.

We have features like funded_amn, int_rate, installment, loan_status and other 49 features as the main features and through the trained model ,we can get the feasibility grade as the result;

For example:

borrower1 input: funded_amn= 19080.1 , int_rate=11.89%, loan_status=Fully Paid…….

output: grade is B

autoML vs manual method

So according to learn and use both sides of ML methods, we know that automl is basically a Pipeline Optimization Tool that wrapped the most tedious part of project which is we don't need to implement the feature engineering ,and they will intelligently exploring thousands of possible pipelines to find the best one for your data.

Reproducibility

During development of a model, sometimes it is useful to be able to obtain reproducible results from run to run in order to determine if a change in performance is due to an actual model or data modification, or merely a result of a new random sample. So we conduct a repeat experiments of manual methods and antoML methods. The autoML methods like TOPT are not as good as manual method like manual methods in the reproducibility, the output score varies a little bit among sevs.eral repeated experiment

Task 5: Analysis

There are still many problems with our data model at the beginning. In practical considerations, there are two reasons why we believe that the model should still not be used in practical applications.

1. The processing of attributes is not perfect, and there are still many variables that do not analyze their characteristics.

   In addition to interest rates, we can analyze the personal credit history to determine the level of credit of this person.

2. The process of feature processing needs to be further improved. We can only use the interface for analysis at present, and the internal logic cannot be understood.

   The results of machine learning are not standard, and different analyses yield a variety of results.