**1. DATASET DESCRIPTION FOR Q1 and Q2**

The LendingClub is a peer-to-peer lending company that directly connects borrowers and potential lenders/investors. In this assignment, you will build classification models to predict whether a loan provided by LendingClub is likely to be a bad loan. In other words, you will use data from the LendingClub to predict whether a loan will be paid off in full or the loan will be charged off and possibly go into default.

We will be using a subset of features (categorical and numeric) from the LendingClub website. The features we will be using are described in the code comments below.

1. 'grade', # grade of the loan
2. 'short\_emp', # one year or less of employment
3. 'emp\_length\_num', # number of years of employment
4. 'home\_ownership', # home\_ownership status: own, mortgage or rent
5. 'dti', # debt to income ratio
6. 'purpose', # the purpose of the loan
7. 'term', # the term of the loan
8. 'last\_delinq\_none', # has borrower had a delinquency
9. 'last\_major\_derog\_none', # has borrower had 90 days or worse rating
10. 'revol\_util', # percent of available credit being used
11. 'total\_rec\_late\_fee', # total late fees received to day
12. target = 'bad\_loans' # prediction target (y) (1 means risky, 0 means safe)

Model the training data "loan\_train.csv" using KNN and CART decision tree receptively. You are required to use the packages distributed in the sample codes of this class for ease of TA grading.

**Question 1**: (3 Marks) Data pre-processing. Several requirements

* 1. For “grade”, treat it as an ordinal categorical variable. Learn how to use the ordinal categorical variable encoder to convert it to numbers.
  2. For the other categorical features, treat them as categorical features and use a onehot encoder to convert those to numerical features.
     1. For a and b, a reference is provided below for you to self-study  
        <https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/>
  3. After a and b, use MinMaxScaler to normalize all features to 0 to 1.
     1. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>
  4. Be careful that a and b should be BEFORE c, so that the ordinal categorical variable is also processed.
  5. **Create a data frame and output the summary statistics of your training data features and test data input features by the describe() function. This is one key step that TA will check your results.**

**Question 2**. (3 Marks) Accuracy, underfitting or overfitting?

1. The goal is to output 2 charts for TA to check. The first chart is to use Tree by using the processed input features from Q1. The second chart is to use Tree without any data pre-processing.
2. In each chart, report training accuracies and test accuracies on the training dataset "loan\_train.csv" and test dataset "loan\_test.csv" respectively. One EXAMPLE of the output is provided below. This example is only to tell you the X and Y-axis, it does not imply the shape is the same.
3. For Tree, the x-axis is “max\_depth”. Also, dt\_depths = [i\*2 for i in range(1, 40)]
   1. The deliverables of Q2 include the following
      1. Two charts.
      2. Identify the best parameter based on the **test dataset**. Report both the training accuracy and test accuracy. So you should report 2 cases: Tree1, and Tree2. In each case, we expect to see the hyperparameter value, and two accuracies.

Chart, line chart

Description automatically generated

Hint: Some Sample codes for generating plots. These are hints and are parts of working codes. These codes alone will not work and you need to modify variable names to fit your own code. But this should save you some time. It is fine that you use your own codes. This piece of code is not a requirement.

# loop to compute test and training accuracy

k\_neighbors = [1, 5, 10, 15, 20, 25, 30]

knn\_models = []

train\_accuracy\_knn = []

test\_accuracy\_knn = []

for n\_neighbors in k\_neighbors:

knn\_model = KNeighborsClassifier(n\_neighbors=n\_neighbors)

knn\_model.fit(X\_preprocd, Y\_train\_procsd)

knn\_models.append(knn\_model)

train\_accuracy\_knn.append(knn\_model.score(X\_preprocd[:something], Y\_train\_procsd[:something]))

test\_accuracy\_knn.append(knn\_model.score(X\_test\_procsd, Y\_test\_procsd))

# plotting figures

plt.figure()

plt.plot(k\_neighbors,train\_accuracy\_knn, c='r', label='Training accuracy')

plt.plot(k\_neighbors,test\_accuracy\_knn, c='g', label='Test accuracy')

plt.xlabel("No. of Neighbors")

plt.ylabel("Accuracy")

plt.legend()

plt.show()

**Learning Objectives for Q3 and Q4**

In this assignment, the learning objectives include the following. First, we will practice hyper-parameter tuning by 2 methods (grid search and random search). Second, you will practice tuning the two most popular packages: XGBoost and LightGBM. Third, you will practice building a stacking classifier for prediction. Last, you will practice a toy version data competition.

**Dataset Description:**

The dataset is from Compustat North America. This is a well-known dataset that includes all accounting numbers of public companies in the USA. This is the standard dataset for professors to conduct research.

The target variable for prediction is quarterly firm-level EPS (EPSFIY: Earnings Per Share (Diluted) - Including Extraordinary Items in Compustat). For constructing the sample of firms, I apply the following two filters.

* USA public companies in only two exchanges: NYSE and NASDAQ
* I only keep companies with no missing value of “epsfiy” from 2008 to 2019 for all quarters.
  + As a result, in the training dataset, we have 2304 companies: 101,376 rows, 44 quarters for each firm.

More about rows of data:

* 2008 to 2017 are your training samples. Just drop 2008 and 2009 since those two years have missing values due to lagged 8 quarters of EPS.
* 2018 is your holdout test dataset (with values of Y) for Questions 3 and 4.
* 2019 Quarter 1 is the holdout test dataset (without values of Y) for our toy data competition Question 4. This dataset is provided in a separate file without the answers (no column EPSFIY).

More about data columns

1. ID: this is a sequential ID. Do not use this in data mining models.
2. Gvkey: this is the firm ID for SEC filing.
3. Tic: This is the ticker that is commonly used in practice. This is just for your reference to Google more about this company (if you are interested in). Note that Ticker is not a firm ID “in database sense”. This is because ticker is similar to company names. The same company may have a different ticker over years (they can change their names) and also different companies may share the same ticker in different years (popular tickers are like popular URLs. When it is available, it will be occupied soon.).
4. Conm: company name for your reference.
5. DataYr: data year
6. DataQtr: data quarter. You can use this as a feature in data competition
7. Period: There are 48 periods in this dataset. This is a sequential number that represents the number of ith quarter in your dataset. You may not need this as a feature in your data mining model. You may use this for finding lagged variables…etc.
8. Epsfiy: This is your Y for prediction.
9. epsfiy.1-8: Last 8 quarters’ Y.
10. NAICS: Industry classification code
11. SIC: industry classification code
    * For the two industry classification codes. Please see <https://www.naics.com/search/> for the naming of each industry.
    * **Do note that these are categorical variables, not an integer.**
    * At the same time, both are hierarchical. For example, SIC 7372 is “Prepackaged Software” and 73 is “Business Services”. So you may need to use the higher-level industry code in Question 5, rather than the full-digtis of NAICS code or SIC code
    * SIC code is an older version of industry classification. For example, all e-commerce companies do not have a category. NAICS is newer but still not perfect. Both codes are widely used in academic papers.

For XGBoost to predict numerical Y, you need to modify and use the following piece of code. I intentionally pick a numerical Y so you can practice both classification and regression prediction in A1.

import xgboost as xgb  
xgb.XGBRegressor(objective ='reg:linear', n\_estimators = 10, seed = 123)

**Questions:**

For panel data prediction, the validation should be more careful than cross-sectional data. For simplicity, we try simple and typical cross-validation in this assignment (while those methods may suffer from information leakage and should be used with caution in the future). We may cover the more complicated times series validation approach later when we cover time-series forecasting models.

For Questions 3 and 4, just use 8 lagged eps as input features. For Q5, you can use any data in the training data file. More about this is provided below.

Q3 (3 Marks): Please train and tune XGBoost by 5-fold cross-validation and two types of parameters tuning methods (grid search and random search). We only tune 2 parameters to save you some time.

* Apply cross-validation on all rows from 2010 to 2017.
* Performance metric is RMSE
* Parameters
  + learning\_rate=0.1 (no tuning, I prefer a smaller value but it will be slower.)
  + n\_estimator from 20 to 200, including 200 and with 30 increments.
  + max\_depth = 3, 4, or 5
* All other parameters are default values. So we try out 18 combinations.
* For random search, we try 60 cases.
  + For max\_depth, 1/3 probability for each of 3 values (3,4,5).
  + For n\_estimators, use equal probability for all integers from 20 to 200.
* Results:
  + Report your best parameter values of 2 cases (grid search and random search). Are the results similar or different?
  + Report the best prediction performance by 2 tuning methods. Also, report the prediction performance of two cases: cross-validation RMSE from 2010 to 2017 and RMSE in holdout sample in 2018.
    - Let me clarify a bit. Your best parameter is decided by 5-fold cross-validation of prediction performance from 2010 to 2017. Asking you to report the performance of 2018 is to help you double-check the prediction performance about whether you overfit (your results are much better from 2010 to 2017, than results in 2018).
  + Set random seeds so TA can verify your results.
  + Please summarize all results into one table or one paragraph so it is easy for TA to read.

Q4: (3 Marks): Modify sample code in Week 5 to build a stacked classifier by XGBoost and LightGBM. The requirements are

* Level-0 has only two classifiers, Random Forest and LightGBM
* Level-1 is the XGBoost classifier. You only tune the number of trees by a loop or grid search and keep all other parameters as the default value.
* For simplicity, you can use all training datasets for cross-validation on both level-0 and level-1.

Q5. (3 Marks): Please build a model to predict earnings in 2019 Q1 test samples.

* You need to submit your training code, prediction model, and predicted value of EPS. Your grading will be mostly based on the prediction performance measured by **RMSE**.
* For prediction performance more than the median of students in this class. you will get 3 out of 3 in Q5.
* For the other half of the class, grading will be based on how poor the performance is and what kind of bugs occur in your coding. Grading 2 or 2.5 is for those submissions slightly below the median (bad luck or slightly poor tuning, such as 25% to 50%). Usually, there are some submissions with very poor prediction performances. Those are usually with mistakes or bugs in the coding and will get <=2 out of 3 in Q5. Don’t over spend your time in this assignment. This is just to help you practice data competition.

**More instructions for Q5:**

* For level-0 and level-1 models, you are allowed to use any combination of models supported in sklearn. You are also allowed to stack more than one classifier from the same algorithm but with different hyperparameters.
* Stacking is optional. You are allowed to use only one classifier without stacking. If you use LightBGM and tune very well, I believe you will still get marks larger than the median.
* You are allowed to tune any parameters of those classifier functions.
* For tuning, you are allowed to use any methods.
* For cross-validation, you are allowed to use any methods (just split train-test by time, any kinds of CV including repeated stratified CV, or you want to self-study or try our other methods is fine too).
* For features engineering,
  + You are allowed to include quarter as an input feature.
  + You should not use the “year” as an input feature. In short, “year” is not allowed to be used as an input feature. This is a mistake.
  + Any features engineering based on this given dataset is allowed. You are not allowed to use additional information outside the given training dataset.
  + You are allowed to explore any feature engineering on Y. For example, you can predict changes in Y q2q or y2y. The idea is once you have the predicted changes (Yt – Y(t-1)), you can easily use the known value of Y(t-1) to derive the predicted value of Yt. This is different from you predict Yt directly. Similarly, you can predict growth % of Yt q2q or y2y.
  + You are allowed to use longer lagged quarters (but you may suffer some missing values in earlier years. You can let XGBoost or LightGBM to handle missing values. But some classification packages may not take missing values).
* For features selection,
  + You are allowed to use any method to drop features. For example, you are allowed to use only lagged 4 quarters as an example.
  + You are allowed to explore any methods for features selection. Let me share with you some additional slides and sample codes for trying if you have time.
* Manage your time well in this assignment. Do not spend too much time on this assignment. That is why I design the grading rule that if your prediction performance is above the median, you get full marks. You don’t need to be the champion in this practice. That is also why I decided not to include more accounting numbers for Q4.
* Surely any kind of cheating is not allowed. In this case, you can find out the EPS of 2019Q1 yourself. You will get 0 for A1 if you get caught in using actual EPS in any way in this exercise. You are allowed to discuss with your classmates and learn from each other. But do not copy the codes and results (parameters of models) from your classmates.