HW5

The LendingClub is a peer-to-peer lending company that directly connects borrowers and potential lenders/investors. In this homework, you will build classification models to predict whether or not a loan provided by LendingClub is likely to default.

In this homework 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.

Load the Lending Club dataset

Load the dataset into a data frame named **loans**.

Exploring features

Let's quickly explore what the dataset looks like. First, print out the column names to see what features we have in this dataset. Here, we should see that we have some feature columns that have to do with grade of the loan, annual income, home ownership status, etc.

Features for the classification algorithm

In this assignment, we will be using a subset of features (categorical and numeric). The features we will be using are described below. Extract these feature columns and target column from the dataset. We will only use these features.

```
features = ['grade',                     # grade of the loan
            'sub_grade',                 # sub-grade of the loan
            'short_emp',                 # one year or less of employment
            'emp_length_num',            # number of years of employment
            'home_ownership',            # home_ownership status: own, mortgage or rent
            'dti',                       # debt to income ratio
            'purpose',                   # the purpose of the loan
            'term',                      # the term of the loan
            'last_delinq_none',          # has borrower had a delinquincy
            'last_major_derog_none',     # has borrower had 90 day or worse rating
            'revol_util',                # percent of available credit being used
            'total_rec_late_fee',        # total late fees received to day
           ]
```

You may skip observations with missing values in these features.

Exploring the target column

The target column (label column) of the dataset that we are interested in is called bad_loans. In this column **1** means a risky (bad) loan **0** means a safe loan.

Now, let us explore the distribution of the target column. This gives us a sense of how many safe and risky loans are present in the dataset. Print out the percentage of safe loans and risky loans in the data frame.

It looks like most of these loans are safe loans (thankfully). But this does make our problem of identifying risky loans challenging.

Sample data to balance classes

Our data is disproportionally full of safe loans. Let's create two datasets: one with just the safe loans (safe_loans_raw) and one with just the risky loans (risky_loans_raw).

One way to combat class imbalance is to undersample the larger class until the class distribution is approximately half and half. Here, we will undersample the larger class (safe loans) in order to balance out our dataset. This means we are throwing away many data points. We will use seed=1 so everyone gets the same results. Use the following codes for this task.

```
# Since there are fewer risky loans than safe loans, find the ratio of the sizes
# and use that percentage to undersample the safe loans.
percentage = len(risky_loans_raw)/float(len(safe_loans_raw))

risky_loans = risky_loans_raw
safe_loans = safe_loans_raw.sample(frac = percentage, random_state = 1)

# Append the risky_loans with the downsampled version of safe_loans
loans_data = risky_loans.append(safe_loans)
```

You can verify now that **loans_data** is comprised of approximately 50% safe loans and 50% risky loans.

One-hot encoding

For scikit-learn's implementation, it requires numerical values for it's data matrix. This means you will have to turn categorical variables into binary features via one-hot encoding.

Split data into training and validation and test sets

First, **s**plit the original data into train-validate set (90%) and test set (10%) using random state = 0.
Then, split the train-validate set into training set (80%) and validation set (20%) using random state = 0.

Build a decision tree classifier

Build a decision tree classifier with max_depth = 2 or 6 or 10 using the training set.
Visualize your decision trees.

Compare the accuracy of the three decision tree classifiers using the validation set. Which one is the best? Report the best model's performance on the test set.

Explore probability predictions

For each row in the validation set, what is the probability (using the best model from the previous comparisons) of a loan being classified as safe? (Hint: if you are using scikit-learn, you can use the .predict_proba() method)

Which loan has the highest probability of being classified as a **safe loan**?

Build a SVM classifier

Use 5-fold GridSearchCV on the train-validate set. Build a kernelized SVM classifier (preprocessing the data with MinMaxScaler) with rbf kernel and compare hyperparameters in the following grid:
C: np.logspace(-3, 3, 7), gamma: np.logspace(-3, 3, 7) / X_train.shape[0]
Report the best model's performance on the test set.

Build a random forest classifier

Use 5-fold GridSearchCV on the train-validate set. Build a random forest classifier and compare hyperparameters in the following grid:
max_features: [2, 4, 6, 8, 10], max_depth: [6, 8, 10, 12, 14]
Report the best model's performance on the test set.

Build a gradient boosting classifier

Use 5-fold GridSearchCV on the train-validate set. Build a gradient boosted tree classifier (fixing max_depth = 6) and compare hyperparameters in the following grid:
learning_rate: np.logspace(-2, 0, 3), n_estimators: [5, 10, 50, 100, 200, 500]
Report the best model's performance on the test set.

XGBoost, LightGBM, and CatBoost

Try XGBoost, LightGBM, and CatBoost and see if you can get further improvement.
Report the best model's performance on the test set.

Final Evaluation

After all the comparisons, which model will you select?