CP322 Machine Learning - Assignment 1
Due Date: Jan 27, 2022 at 11:59 PM

About Submission

When writing and submitting your assignments follow these requirements:

- You are expected to submit a single IPython Notebook file for this assignment.
- Late assignment submissions will not be accepted and will be marked with 0.
- Your assignment should be submitted online through the MyLearningSpace website. Email submission is not accepted.
- Please document your program carefully.

Before You Start

0.1 Setting-up Your Software Environment

This part simply requires you to setup the programming environment that we will be using for the remainder of the course. As stated in class, you may install the required software in your personal computer, in which case we suggest you carefully ensure that everything you install is up-to-date, which will help avoid compatibility issues when your assignments are graded.

Programming assignments will require the use of Python as well as additional Python packages. Most of the relevant software is a part of the SciPy [1]stack, a collection of Python-based open source software for mathematics, science, and engineering (which includes Python, NumPy, the SciPy library, Matplotlib, pandas, IPython, and scikit-learn). The Anaconda Python Distribution[2] is a free distribution for the SciPy stack that supports Linux, Mac, and Windows. Ensure that your machine has the following software installed:

- Python (An interactive, object-oriented, extensible programming language.)
- NumPy (A Python package for scientific computing.)
- SciPy (A Python package for mathematics, science, and engineering.)
- Matplotlib (A Python package for 2D plotting.)
- pandas (A Python package for high-performance, easy-to-use data structures and data analysis tools.)
- IPython (An architecture for interactive computing with Python.)
- scikit-learn (A Python package for machine learning.)

0.2 Create Your First Notebook

To create a new IPython Notebook, you simply need to open the terminal *Jupiter Notebook*, and this will bring up the IPython web interface from where you may select *New Notebook*. Once you are finished, rename and save your assignment, and this will generate an .ipynb file.

1 Decision Tree

In this assignment, you will use the *scikit-learn*'s decision tree to predict the risk of lending money. The aim of this question is for you to read the *scikit-learn* API and get comfortable with exploring basic statistics, developing classification models, and handling training/validation splits.

[1] https://www.scipy.org/
[2] https://www.anaconda.com/distribution/

1.1 Know Your Data: 5 points

We will explore a publicly available dataset from *LendingClub*[3], which connects people who need money with people who have money. We attempt to construct a model to analysis the risk of lending money to people given various profile data. In particular, we exploit the historical data to predict **whether or not the borrower paid back their loan in full** (*not.fully.paid* is the target classification label, whereas *credit.policy* is a descriptive feature). Here are the meanings of different columns in the data set:

- credit.policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.
- purpose: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").
- int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates.
- installment: The monthly installments owed by the borrower if the loan is funded.
- log.annual.inc: The natural log of the self-reported annual income of the borrower.
- dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income).
- fico: The FICO credit score of the borrower.
- days.with.cr.line: The number of days the borrower has had a credit line.
- revol.bal: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).
- revol.util: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).
- inq.last.6mths: The borrower's number of inquiries by creditors in the last 6 months.
- delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.
- pub.rec: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).
- not.fully.paid: The quantity of interest for classification - whether the borrower paid back the money in full or not

Please complete the following tasks:

1. Print the first 5 records of your data.
2. Demonstrate the basic statistics of different features, i.e., count, mean, std, min, max, and 25:50:75% percentiles.
3. Show the breakdown of credit approval status (reflected in *credit.policy*). In our original data, 1 indicates "approved", 0 means "not approved".
4. In a single plot, draw the histogram of installments by "approved" and "not approved", where *bins=35*.
5. Illustrate the boxplots of Fico score (reflected in *fico*) that varies between "approved" and "not approved" borrowers.

1.2 Data Preprocessing and Model Construction: 6 points

1.2.1 Data Preprocessing and Splitting

The "purpose" feature in our dataset takes nominal values, including "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other". Please convert this feature with dummy variables that *scikit-learn* can recognize. More specifically, it can be expanded into 6 different new features, with each indicating whether a special purpose is served with a boolean value.

To evaluate the effectiveness of your method, the given data should be split into two parts. In this question, a ratio of 70%: 30% is set between training and testing data, i.e., you need to randomly select 70% of the data as training, and leave the rest as testing data.

[3]https://www.lendingclub.com/

1.2.2 Training a Decision Tree

- Using the preprocessed **training** data to construct decision trees for decision making with the help of *DecisionTreeClassifier*[4]. Two splitting criteria should be tested, i.e., 1) Information Gain, 2) Gini coefficient. Such criterion will be set in the function call as a hyper-paramter.
- Using a 10-fold cross-validation[5] to identify which criterion gives the best performance. Note that the original training data will be further divided into a training and a validation dataset for cross-validation. The simplest way to use cross-validation is to call the cross_val_score helper function on the estimator and the dataset.

Figuring out how to use this implementation, its corresponding attributes and methods is an important part of the assignment.

1.2.3 Training a Random Forest

- Using the preprocessed training data, construct two random forests[6] based on information gain and Gini respectively. Each random forest needs to fit 15 decision trees on 1000 sub-samples of the training dataset. The sub-sample size in *scikit-learn* is controlled with the $max\_samples$ parameter if $bootstrap = True$, otherwise the whole dataset is used to build each tree.
- Using a 10-fold cross-validation to identify which criterion gives the best performance.

1.3 Performance Evaluation and Analysis: 4 points

To compare the performances of the classifiers constructed above, examine your models with the **testing** data using three evaluation metrics, including precision, recall, and f-score. Finally, to conclude your work, please use **concise language** to analyze the results based on your observation.

[4]https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[5]https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
[6]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier