**CS405 Machine Learning**

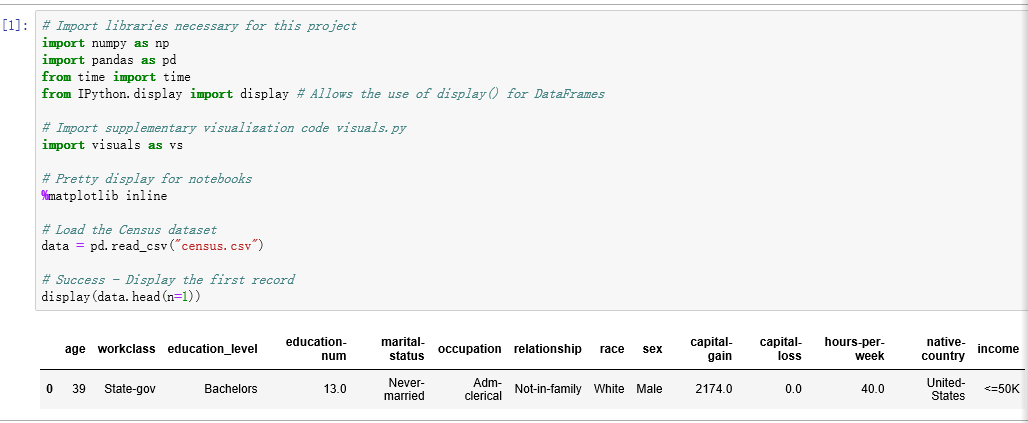
**Lab #0 Preliminary**

**Lab (75 points)**:

**Objective**s：This lab will introduce how to pre-process and transform data to make machine-learning algorithm work. In this lab, you will employ several supervised algorithms of your choice to accurately model individuals’ income using data collected from the 1994 U.S census. Your goal with this pre-lab is to construct a model that accurately predicts whether an individual makes more than $50000.

**Exploring the Data**

Run the code cell below to load necessary Python libraries and load the census data. Note that the last column from this dataset “income”, will be our target label (whether an individual makes more than, or at most, $50,000 annually). All other columns are features about each individual in the census database.



**Exercise 0:**

A cursory investigation of the dataset will determine how many individuals fit into either group, and will tell us about the percentage of these individuals making more than $50,000. In the code cell below, you will need to compute the following:

* The total number of records, ‘n\_records’;
* The number of individuals making more than $50000 annually, ‘n\_greater\_50k’.
* The number of individuals making at most $50000 annually,‘n\_at\_most\_50K’.
* The percentage of individuals making at more than $50000 annually, ‘greater\_percent’’
* Feature values for each column

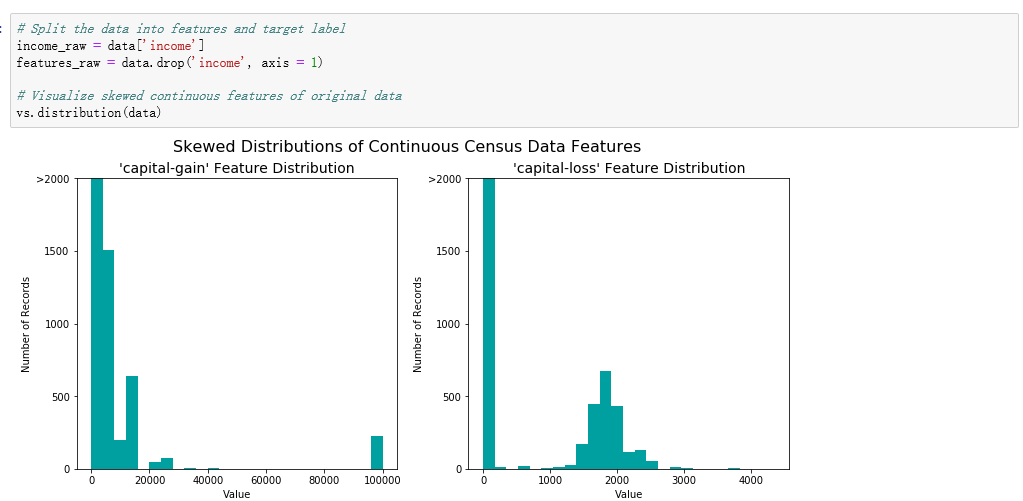
Tips :As the data is stored as pandas, this tutorial will help you finish <https://pandas.pydata.org/pandas-docs/stable/10min.html>

**Preparing the Data**

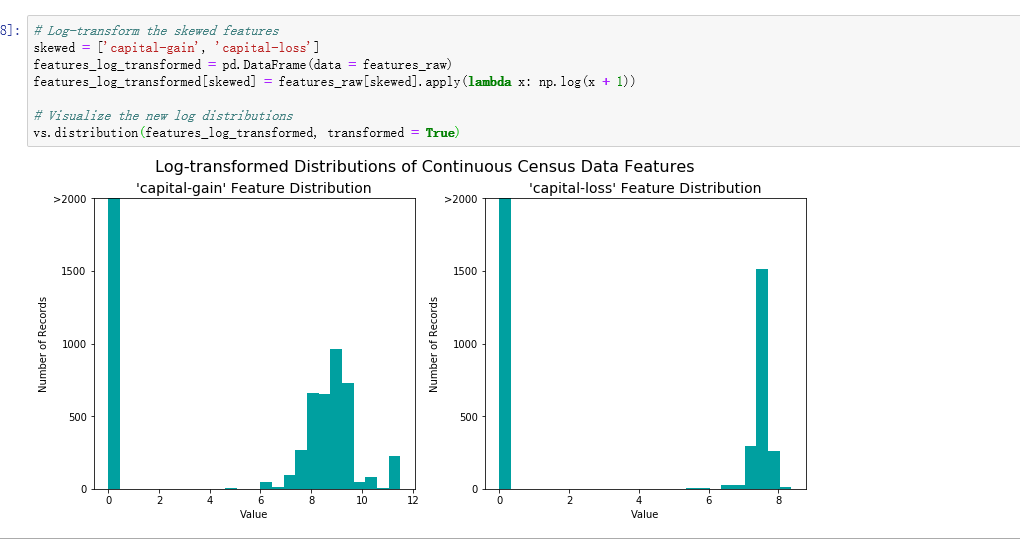
Before the data can be used as the input for machine learning algorithms, it often must be cleaned, formatted, and restructured — this is typically known as preprocessing. Fortunately, for this dataset, there are no invalid or missing entries we must deal with, however there are some qualities about certain features that must be adjusted. This preprocessing can help tremendously with the outcome and predictive power of nearly all learning algorithms.

**Transforming Skewed Continuous Features**

A dataset may sometimes contain at least one feature whose values tend to lie near a single number, but will also have a non-trivial number of vastly larger or smaller values than that single number. Algorithms can be sensitive to such distributions of values and can underperform if the range is not properly normalized. With the census dataset two features fit this description: 'capital-gain' and ‘capital-loss'. The code cell below will plot a histogram of these two features. Note the range of the values present and how they are distributed.



For highly-skewed feature distributions such as ‘capital-gain' and ‘capital-loss’, it is common practice to apply a logarithmic transformation on the data so that the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers. Care must be taken when applying this transformation however: The logarithm of 0 is undefined, so we must translate the values by a small amount above 0 to apply the the logarithm successfully. Below code cell will perform a transformation on the data and visulaize the results. Again, note the range of values and how they are distributed.



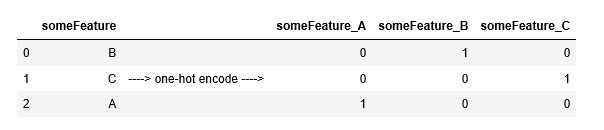
**Normalizing Numerical Features**

In addition to performing transformations on features that are highly skewed, it is often good practice to perform some type of scaling on numerical features. Applying a scaling to the data does not change the shape of each feature's distribution (such as ‘capital-gain’ or ‘capital-loss’ above); however, normalization ensures that each feature is treated equally when applying supervised learners. Note that once scaling is applied, observing the data in its raw form will no longer have the same original meaning, as exampled below.

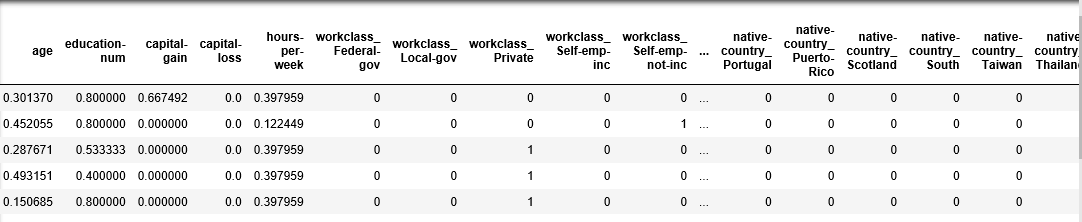


**Data Preprocessing**

From the table in above, we can see there are several features for each record that are non-numeric. Typically, learning algorithms expect input to be numeric, which requires that non-numeric features (called ‘categorical variables’) be converted. One popular way to convert categorical variables is by using the one-hot encoding scheme. One-hot encoding creates a ‘dummy’ variable for each possible category of each non-numeric feature. For example, assume some features has three possible entries:A,B or C We then encode this feature into someFeature\_A, someFeature\_B and someFeature\_C.



Additionally, as with the non-numeric features, we need to convert the non-numeric target label,’income’ to numerical values for the learning algorithm to work. Since there are only two possible categories for this label ("<=50K" and ">50K"), we can avoid using one-hot encoding and simply encode these two categories as 0 and 1, respectively.



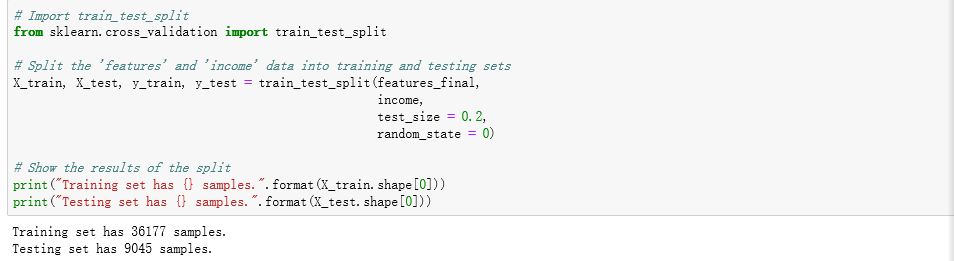
**Exercise 1:**

* **Perform one-hot encoding on the data**
* **Convert the target label ‘income\_raw’ to numerical entries (set records with “<=50k” to 0 and records with “>50k” to 1).**

**Shuffle and Split Data**

Tips: pandas.get\_dummies() can perform one-hot encoding.

When all categorical variables have been converted into numerical features, and all numerical features have been normalized. As always, we will now split the data (both features and their labels) into training and test sets. 80% of the data will be used for training and 20% for testing.



**Evaluating Model Performance**

**Accuracy** measures how often the classifier makes the correct prediction. It’s the ratio of the number of correct predictions to the total number of predictions.

**Precision** tells us what proportion of messages we classified as positive. It is a ratio of true positives to all positive predictions. In other words,

Precision = TP/(TP + FP)

**Recall(sensitivity)** tells us what proportion of messages that actually were positive were classified by us as positive.

Recall = TP/(TP + FN)

We can use **F-beta score** as a metric that considers both precision and recall:

**Exercise 2**: Now we chose a model that always predicted an individual made more than $50,000, what would that model's accuracy and F-score be on this dataset? You must use the code cell below and assign your results to 'accuracy' and 'f-score' to be used later.

Tips: the following are some of the supervised learning models that are currently available in *scikit-learn* that you may choose from:

* Gaussian Naive Bayes (GaussianNB)
* Decision Trees
* Ensemble Methods (Bagging, AdaBoost, RandomForest)
* K-Nearest Neighbors
* Support Vector Machines (SVM)
* Logistic Regression

**Questions:**

1. An important task when performing supervised learning on a dataset like the census data we study here is determining which features provides the most predictive power. Choose a *scikit-learn* classifier (e.g adaboost, random forests) that has a feature\_importance\_ attribute, which is a function that ranks the importance of features according to the chosen classifier.

List three of the supervised learning models above that are appropriate for this problem that you will test on the census data.

1. Describe one real-world application in industry where the model can be applied
2. What are the strengths of the model; when does it perform well?
3. What are the weaknesses of the model; when does it perform poorly?
4. What makes this model a good candidate for the problem, given what you know about the data?

**CS405 Machine Learning**

**Lab #1 Decision tree**

**Pre-lab (25 Points)**

**Objective**s：In this exercise, you will implement the decision tree andobserve its results. Before starting on this programming exercise, it is recommended that you have finished the lab above to learn related training and testing procedures.

**Overview**

Decision Tree Analysis is a general, predictive modeling tool that has applications spanning a number of different areas. The decision rules are generally in form of if-then-else statements. The deeper the tree, the more complex the rules and the fitter the model.

From the above lab, it is assumed that you have got familiar with some of the terminologies:

* Instances: refer to the vector of features or attributes that define the input space.
* Attribute: A quantity describing an instance
* Concept: The function that maps input to output
* Target Concept: the function that we are trying to find the actual answer
* Hypothesis Class: set of all the possible functions
* Sample: A set of inputs paired with a label, which is the correct output
* Candidate Concept: A concept which we think is the target concept
* Testing set: test the candidate concept and determine its performance

**Exercise**

1. Given the training dataset “train.txt” and testing dataset ‘test.txt’, implementing decision tree algorithm in scikit-learn to see how it performs. (The last column of each data is regarded as the label)
2. Write the pseudocode of decision tree classifier
3. Writing your own code about decision tree classifier by python or MATLAB, plot your decision tree classifier
4. compare the predicting results between *scikit-learn* classifier and yours.