CMPUT 466 Mini-project Initial Draft

Ziming Fu 1370023
Jiazhen Li 1501597

Introduction:

Surveys have shown that both in Canada and the U.S. It's the same reason why politicians love the word so much: When they saying "middle class," the vast majority of voters think it's about them, even though it might not be.

So what are some of the features shared by the population in the middle class? The median household income reached $57,617 in 2018, according to the latest census data. Our study is to determine whether a person's income exceeds 50k/year (the lower threshold for the entry of middle class) depends on the observation of features. This is a typical binary classification problem. Three different machine learning algorithms (Logistics regression, SVM, and Naive Bayes) with different hyper-parameters were studied with the project. We will analyze the general performance of these algorithms and evaluate the statistical significance of the following tests.

Dataset description:

Our dataset is a real-world "Census Income" dataset, collected and donated by UCI Data Mining and Visualization Silicon Graphics. More details on the resource link.
The dataset is a census data with 14 features and a target label: y ⇒ indicates whether a person's income exceeds 50k/year or not? (Binary label: "1" means yes; "0" means no).
There are two main types of features: categorical features and numerical features. The dataset contains a total of 48842 instances with some missing data scattered.
The specification of the features are listed:

age	continuous.
work-class	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, etc.
fnlwgt	continuous.
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, etc..
education-num	continuous.
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, etc.

occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, etc.
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex	Female, Male.
capital-gain	continuous.
capital-loss	continuous.
hours-per-week	continuous.
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, etc.

Knowing issues:

- Handling data rows with missing value:
    1. Simply delete the samples from the dataset.
    2. Replace the missing value with None.
    3. If the missing value is category type, replace it with the most frequent instance from the category; If the missing value is numerical type, replace it with the mean/median of the distribution.

Approaches and Learning Methods:

Three approaches are used for this project: Logistics regression, Support vector machine(SVM), and Naive Bayes.

- **Logistic regression**
    - Description & why appropriate:
        - Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.
        - $$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$
    - Implementation:

        - Logistic regression is implemented via importing the scikit learn lib -- LogisticRegression. This implementation can fit the binary, One-vs-Rest, or multinomial logistic regression with optional, or Elastic-Net regularization.
    - Hyper-Parameters:
        - Stepsize (learning rate): 0.001, 0.01, 0.1.
        - Weight decay

- **Support vector machine(SVM)**
    - Description & why appropriate:
        - The support vector machine is a discriminative classifier formally defined by a separating hyperplane. A hyperplane is a line that divides the input variable space. In the SVM, the hyperplane is selected to optimally separate the points in the input variable space from their classes (0 level or 1 level). The SVM learning algorithm finds the coefficients that cause the hyperplane to best separate the classes using kernel trick.
        - SVM perfectly works with binary classification problems. And with many features this dataset, the benefit is that we can capture much more complex relationships between data points without having to perform difficult transformations on our own. SVM is effective in high dimensional spaces. Also, it uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
    - Implementation:
        - Support vector machine(SVM) is implemented via importing the scikit learn lib. The support vector machines in scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data.
    - Hyper-Parameters:
        - Different kernels: linear, polynomial, and radial.
        - Weight decay
        - Margin

- **Gaussian Naive Bayes:**
    - Description & why appropriate:

    - Gaussian Naive Bayes estimates a probabilistic model of the classification problem as described below. The prior for the coefficient is given by a spherical Gaussian. It is a supervised learning algorithm based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.
    - Our target variable is discrete so we can use Naive Bayes methods. And since our dataset has continuous features, we shall use gaussian distribution for p(xi|y) instead of Bernoulli distribution. Also, It can be used to include regularization parameters in the estimation procedure.
    - $$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$
    - Also, Naive Bayes is the most straightforward and fast classification algorithm, which is suitable for a large chunk of data.
  - Implementation:
    - Gaussian Naive Bayes is implemented via importing the scikit learn lib - The Gaussian Naive Bayes building in the classifier.
  - Hyper-Parameters:
    -

Experiment Design:

- **Data handling:**
  - **Data exploration:**
    - Evaluate the numbers of data chunks separated by 2 different labels of the target column ie. 1 or 0. If there is a big difference in terms of size between the two numbers' percentage weight, the classes need to be balanced.
    - To balance the classes, we can simply create dummy copies from the minor class to up-sample them. Or we could use the SMOTE algorithm(Synthetic Minority Oversampling Technique) which chooses one of the k-nearest neighbors and uses it to create a similar but randomly tweaked new observations.
    - NOTE: Only up-sample data in the training set!

- **Training-test sets splitting:**
    - Typically, we split the dataset into 3 parts: training set, validation set, and test set. However, by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.
    - **K-fold cross-validation** was introduced to solve such a problem above and also prevent overfitting and generalize the model to seen data with low bias and variance. In *K*-fold cross-validation, the data is partitioned into *K* subsets. We then iteratively train on each set of (*K-1*) subsets and validate on the remaining subset, allowing each data point to be used as training data (*K-1*) times and validation data 1 time. (If the data is imbalanced, balance only the training set before each training)The model score is then computed as the average validation score across all *K* trials.
    - **K = 10** folds for cross-validation is a good choice for this problem.

- **Training Validating and Testing:**

  #TODO

- **Meta-parameter tuning and selection techniques:**
    - When evaluating different algorithms, or hyper-parameters, it is common to use **k-fold cross-validation** as mentioned in training -test datasets splitting. This allows you to train and test k models with different hyperparameters across the algorithms and collect their evaluation metrics, using their mean as an indication of the performance of the algorithm.

- **Statistical significance tests**:
    - Null Hypothesis:
      Our null hypothesis is that the performance metrics are equal, that any small gain, or loss, observed is not statistically significant.
    - Appropriate Statistical Test:
      Wilcoxon's signed-rank test

Since we need to compare 3 algorithms, it is better to make significant tests pairwise to reject H0 more clearly. Wilcoxon's signed-rank test is a good choice for this since it is a non-parametric test and the distribution of the differences between the two samples cannot be assumed to be normally distributed. Scipy includes an implementation of the Wilcoxon signed-rank test in Python so we implemented Wilcoxon's Signed Rank test from scipy.

- **Models Results Comparison**:
  - Check the Significant test result above. Use confusion matrix to calculate accuracy, precision, recall.
  - Accuracy:
  - Precision:
  - Recall:
  - F1-score:
  - F1 function: F1=(2×Precision×Recall)/(Precision+Recall).
  - Draw ROC(Receiver Operating Characteristic) curve: the two main indicators in the ROC curve areReal rate and false-positive rate.
  - Calculate AUC(area under the curve).

"Winner" algorithm & analysis:

#TODO

References:

#TODO