Kevin Pfeifle and Nolan Meeks
CS 484-001
Final Project Report

Description:

Our project is based on the "Adult Data Set" from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Adult). The goal of the project is to "predict whether income exceeds $50K/yr based on census data". We accomplished this goal by using a variety of different classification techniques in combination with this dataset.

This dataset was extracted from a 1994 census database, and the data is used to determine whether or not a person listed in the dataset falls into one of two categories (we later make these categories binary for simplicity sake). These two categories are income > $50k and income <= $50k. The dataset has a large number of attributes associated with each entry, some of which are not as useful as others. This dataset has 48842 instances, each instance has 14 (15 if you include the income) attributes, and there are missing values in some of the instances. The dataset is also imbalanced, and has about a 75%-25% split across the two categories. The data is broken down into two data files – "adult.data" and "adult.test".

The dataset's attributes are described in the table below. It includes the name of the attribute, a description of what the data represents, and the various values the data can take on.

Name	Description	Data-type
age	The age of the person	continuous
workclass	The workclass of the person	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
fnlwgt	Weights on the current population survey	continuous
education	The person's highest education level	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
education-num	An integer representing the person's highest education level	continuous
marital-status	The person's marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	The category of the person's occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship	The person's current relationship status	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race	The race of the person	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	The sex of the person.	Female, Male
capital-gain	The net capital-gain of the person	continuous
capital-loss	The net capital-loss of the person	continuous
hours-per-week	The number of hours the person works per week.	continuous
native-country	The native country of the person.	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands
income	Whether or not the person makes more than 50k annually.	>50k, <=50k

As you can see, each entry in this dataset has a very large number of potential attributes that it can contain and as a result some of the data is either not useful to us for the purposes of classification, or it is a bit redundant. In particular, the "fnlwgt "attribute is useless to us for the purposes of classifying by income. It gives no information in regard to classifying an instance's annual income. Additionally, the "education" attribute is of little use to us as well – it contains a complicated number of string attributes that is better represented by the "education-num" attribute. The "education-num" attribute condenses all of that information down into an integer to represent the number of years of education. With this in mind, during our implementation, we remove these two attributes from our data during the pre-processing phase.

Methods:

Our method of implementing this classification is broken down into a number of steps: preprocessing, data preparation, and classification. Each of these steps went through a number of iterations to get it to the state that it is currently in, and we believe that we have the best current state that we could achieve within the time-frame of the assignment. We used Python to implement all aspects of our project, and we classified the data based on the KKN classifier, the Naïve Bayes classifier, the Decision Tree classifier, and Support Vector Machines.

Kevin Pfeifle and Nolan Meeks
CS 484-001
Final Project Report

Pre-processing:

Our pre-processing steps have take a variety of forms, the first few of which we mistakenly attempted to do so from scratch. We at first stated reading in the data provided by "adult.data" and "adult.test". To do this we ran a number of loops to iterate over the data and trim out the parts that we did not want. We also performed operations to cast the continuous data into integers and floats as necessary, and after a number of iterations and a lot of code we finally had a clean data set to work with.

After that preprocessing was done the data was easy to read and manage, however it was not in a form that could be passed to our classifiers, so we had to find a way to encode the data. This was an issue though, because we could not find an encoder that would be able to encode the data in the form we had it in, and encoding it ourselves would be to difficult due to the number of different values attributes can take on. We then decided we needed to try a different method of pre-processing the data.

We came across the "pandas" module for python after a bit of research, and found that it had a large number of functions that accomplished *exactly* what we were trying to do, and therefore we completely restructured our pre-processing section to make use of a number of "pandas" functions. We found a read_csv() function in "pandas" and after looking at the data provided we realized that the files are actually csv files even though the extension was not csv. So, we reformatted the data files to be csv files and used "pandas" to read in the data – this trimmed down our code by around 20 lines. We also then used simple and well-known Python methods of removing and replacing data in lists to remove the attributes we don't need, and to change our income attribute to be binary to simplify the classification process.

Data Preparation:

In our first iteration of the project that was discussed in the previous section, we did not make it very far into this section. We attempted to encode the data, but had complications because we found encoders that would encode string data, however it ran into problems with continuous data. For a while we were stuck on this and even attempted to make our own encoder, but then we found that "pandas" had a built-in encoder that had the same functionality that we were attempting to create. Therefore, we elected to scrap the work we had previously done in exchange for a tried-and-tested function that would be guaranteed to work with our classifiers.

After making the change to our updated version, the data preparation stage became quite simple and frankly much less confusing code to read. It simply removes the income field from the data set and turns it instead to labels – since we are classifying based on this field it is important that we are not allowing the value to be used in the classification steps. Then, we used the get_dummies() function from "pandas" to encode all of our data so that it is in a form the classifiers will accept. Then we updated a slight error that occurred in the encoding to fix the value (this took us quite a while to realize and had been throwing off our results until we understood the issue). Finally, we ensured the data in both sets was in the correct order, because if it wasn't then we would get incorrect results.

Kevin Pfeifle and Nolan Meeks
CS 484-001
Final Project Report

Classification:

We used a number of different classifiers to classify our data in an attempt to get the best possible classification that we could. We stuck to classifiers that were discussed directly in the class slides on classification, therefore we had a bounded scope to stick to that we would be working towards. We used scikit-learn for all of the classifiers, and the code for each is quite standardized – we create an object for the classifier, fit the training data and labels to the classifier, and then use the sklearn mean_absolute_error() to determine the absolute error between the expected labels of the test data and the generated labels created by the classifier.predict() function.

Decision Tree Classifier

This classifier was implemented quite simply by using scikit-learn's DecisionTreeClasifier, and using the entropy version. We then simply followed the steps laid out above and got our result.

Naïve Bayes Classifier

This classifier took a bit more work to get functional. We began doing it the standardized way, and found that the error was quite high. So, we decided to perform undersampling on the dataset as well, for this classifier only, since there is a 75%-25% split of the data. This classifier is the only one where the split had any sort of impact, so we only used undersampling for this one. After using imblearn's RandomUnderSampler, we managed to get our results to a slightly better accuracy.

K-Nearest Neighbors Classifier

We actually performed this classifier in three different ways. The code is simple and follows the standardized way discussed above. However, we performed the classification three times, each with a different value for k. We used k = 1, k = 3, and k = 5 respectively.

Support Vector Machine

This classifier was simple and followed the standardized method described above.

In addition to the steps that we took to achieve our classifications, we had wanted to perform some sort of cross-validation. However, due to our data processing and the form of data that we had, we were unable to get it working. Though, because of how we implemented the labels in our system we had a form of accuracy checking by comparing against the labels that we knew were correct, and therefore we feel that it is okay that we don't have cross-validation.

Results:

Since we used a number of different classifiers in order to get the best accuracy we therefore had a different result for each one. The one with the best accuracy, support vector machine, is the value used for all of our interpretation of the dataset.

Decision Tree Classifier – A decent classifier that is mostly accurate.

- Mean Absolute Error: 0.13911921872120878
- Accuracy Score: 86%

Naïve Bayes Classifier – Not that great of an accuracy score.

- Mean Absolute Error: 0.24167966718668746
- Accuracy Score: 76%

K-Nearest Neighbors Classifier (k = 1) – An okay accuracy score, better than naïve bayes.

- Mean Absolute Error: 0.18125422271359254
- Accuracy Score: 82%

K-Nearest Neighbors Classifier (k = 3) – An okay accuracy score, better than k = 1.

- Mean Absolute Error: 0.15969535040845156
- Accuracy Score: 84%

K-Nearest Neighbors Classifier (k = 5) – An okay accuracy score, better than k = 1 and k = 3. We found a trend of accuracy increasing as k increased up until around 5. After k = 5, the accuracy began to go down.

- Mean Absolute Error: 0.1487623610343345
- Accuracy Score: 85%

Support Vector Machine – The best classifier, certainly not always accurate, but it is accurate most of the time.

- Mean Absolute Error: 0.13088876604631164
- Accuracy Score: 87%

Using our best classification data, we attempted to analyze the dataset to find any trends amongst the different data attributes. We found a strong correlation between the hours worked per week and highest education that have person has achieved, in regard to their income. On average, those who make more than $50k annually work significantly more hours per week than those that make less than $50k annually. In addition, we found that the more education a person has, the more that they tend to work per week. Lastly, we found that on average men work more hours per week at lower education levels than women, and at higher education levels both sexes work about the same amount per week.

All of these claims can be visualized on the graph that we created below. The x-axis maps to an integer representing the highest education received, the y-axis maps to the number of hours worked per week. Blue is men, orange is women. The left graph is annual income less than $50k, and the right graph is annual income greater than $50k.

We managed to find a few other different correlations as well, and tried to make graphs on them. But after graphing them we found that the correlations were not as strong as we had thought, and we decided not to include them in the report.

Conclusion:

Throughout this project we used the adult data set from the UCI Machine Learning Repository in order to accurately classify census data into two categories, those who make less than $50k annually, and those that make more than $50k annually. We did this by processing our data, and then using the training data with different classifiers that would classify our test data into those two categories. Finally, we used each of the main classifiers that we used in class, and established which was best for this dataset.

After classifying this data, we feel close to the dataset, and we were able to do some analysis on it. This was an interesting project that gave us more knowledge on the process of classifying datasets.