Project 2

Classification of the data using various classification methods

Team Name: | Prathamesh Pawar | Ved Contractor| Vishruti Patel

Date: 05/03/2018

California State University, Fullerton



Team members information

Prathamesh Pawar

[ppawar@csu.fullerton.edu](mailto:ppawar@csu.fullerton.edu)

Ved Contractor

[helloved@csu.fullerton.edu](mailto:helloved@csu.fullerton.edu)

Vishruti Patel

[vishruti123@csu.fullerton.edu](mailto:vishruti123@csu.fullerton.edu)

# Software used

Like previous project we used Python language for solving the assigned task. Python being very compact language has very efficient syntax. Classification is easy with Python because Python supports numeric. There are several libraries available for manipulation of data in Python which makes it easy and fast to provide solution to the problem like this.

For this project we used Pandas library to read and manipulate data from csv file. There are other libraries like NumPy and Scikit learn for numeric operations and machine learning algorithms. Also, there is provision to plot the graphs which is used using matplotlib library. There is one more similarity between Python and MATLAB. Both these support array manipulations.

OCTAVE/MATLAB are the alternatives that can be used instead of Python for the numeric manipulation. R is a popular language for these kinds of problems as well. Though that may be many features from these languages are ported to Python already. As usual we used **Pandas** library to read the dataset provided in the problem statement. This dataset is manipulated using **Pandas** as well. **Scikit-learn** package is used for preprocessing the data, model selection and for importing all the three classifiers that we will be using in this project.

# Methods used

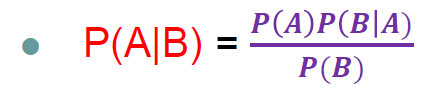
In the problem statement the dataset provided have number of attributes. Attributes given are Age, Work-class, Education level, years of education, Marital status, Occupation, Relationship, Gender, Capital gain, Capital loss, Hours per week and Income. The problem statement asks us to find out the attributes of the people whose salary is above 50,000 per year. We have used mainly three classifiers for this project. They are Naïve Bayes, Decision tree induction and Multilayer Perceptron Classifiers.

## Gaussian naive bayes:

Naïve Bayes is the algorithm for classification in machine learning. This algorithm is simple probabilistic classifier. This algorithm is based on the Bayes theorem. Bayes theorem in probability theory describes the probability of an event, based on prior knowledge of conditions that might be related to that event. For example, if the diabetes if related to age, then, using Bayes’ theorem, a person’s age can be used to more accurately estimate the probability that they have diabetes compared to the assessment of the probability of the diabetes made without knowing person’s age.

Here we are using Age, Work-class, Education level, years of education, Marital status, Occupation, Relationship, Gender, Capital gain, Capital loss, Hours per week and Income attributes to estimate the probability of a person to make over $50,000 a year.

**Bayes rule:**



This rule replaces the joint probability of the event occurring with the conditional probability. That means if an event A occurs there are certain chances of event B occurring can be given by above formula. This formula only gives probability for 2 attributes but we are going to use all the attributes mentioned above to estimate probability of the person to make $50,000 a year.

DECISION TREE CLASSIFIER:

The Decision Tree classifier is a non-parametric supervised learning method. It aims to create a model which could predict the value of a targeted variable by learning simple decision rules inferred from the data set.

A decision tree can be used for decision making through classification. Basically, Decision Tree classifier is a systematic way to make decisions regarding the set of data available. Decision Tree is one of the most visualized classifier, i.e. easy to understand. It can perform multiple classification on a data set.

Once the decision tree has been constructed, classifying a test record is straightforward. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test. It then lead us either to another internal node, for which a new test condition is applied, or to a leaf node. When we reach the leaf node, the class label associated with the leaf node is then assigned to the record, it traces the path in the decision tree to predict the class label of the test record, and the path terminates at a leaf node labeled YES/NO.

## Multilayer perceptrons:

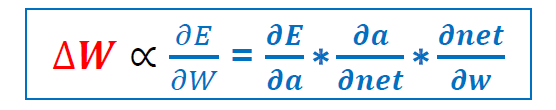
Multilayer perceptron is a class of feedforward artificial neural network. This method uses supervised learning technique. The method used is called Backpropagation. This method uses artificial neural networks to in calculation of the weights to be used in networks by calculating the gradient. This method is special case of automatic differentiation. In this method error is calculated at the output and distributed back through the network layers. For this reason, it is also called backward propagation of errors. This method falls under supervised learning.

Supervised learning is each example is a pair of input object and desired output value. A supervised learning algorithm analyzes the input objects and desired output values and produces a function. This function can be used to map new examples. To map new examples or to predict the outcomes or probabilities the algorithm needs to generalize.

This multilayer perceptron is different from linear perceptron in a way that multilayer perceptron has multiple layers and non-linear activation. Multilayer perceptrons are also called vanilla neural networks if they have single hidden layer.

Here we are using multilayer perceptron to reduce multiple attribute layers to two-layer input-output model to map weighted inputs to map the output of each neurons.

**Chain Rule**



We are going to use this chain rule to find rate of change output to error. For this we have module called *neural\_network*of the *Sklearn* package.

## **Comparison and analysys**

1. Gaussian naïve bayes:

After splitting the data into training and testing set, the model fitting is done and predictions are made. We get three different values as input for our program is 3 for k-fold method. The calculated values are [0.7962046204620462, 0.8001925192519251, 0.7997524412047862].

The mean accuracy obtained for this classification is ***0.8058812581425647***

2. Decision Tree Classifier:

After splitting the data into training and testing set, the model fitting is done and predictions are made. We get three different values as input for our program is 3 for k-fold method. The calculated values are [0.8481848184818482, 0.8507975797579758, 0.8484390042635126]

The mean accuracy obtained is ***0.8514796203238414***

3.Multilayer Perceptron:

After splitting the data into training and testing set, the model fitting is done and predictions are made. We get three different values as input for our program is 3 for k-fold method. The calculated values are [0.8194444444444444, 0.8147689768976898, 0.7754091596754229]

The mean accuracy we obtained is ***0.834356970035362***

As seen from above, the mean accuracy for *Decision Tree classifier* is *highest* out of all the three and hence, it is the best classifier.

## **Brief report OF the Project:**

As the initial step, we imported all the basic libraries and methods necessary to run the program. Then we called the given dataset and performed basic operations to analyze the dataset using methods like. head(), .describe(), .info(). We then calculated the correlation among few parameters and plotted the correlation heatmap. We then performed data cleaning in which we assigned the value 0 for salary <=50K and 1 for salary >=50K. Further, we converted variables to the numeric values to make the compilation easy. Now the program is ready to be splitted in training set and testing set. For our dataset the target value or data is "Salary" and the rest are Feature data. The data was then splitted to training and testing set using train\_test\_split method. We also used K- fold cross validation method. validate the data.

Classification is performed using three different classification:

1. Gaussian Naive Bayes Classifier.

2. Decision Tree Classfier.

3. Multi-Layer Perceptron Classifier:

Using the methods GaussianNB(), DecisionTreeClassifier() and MLPClassifier() for each classifiers respectively, the model was fitted and predictions were made. We got three values one for each folds as we used 3 folds while coding and then calculated the mean accuracy for each of the classifiers. And at the end, as per the requirement of the project, the profile of all the persons having salary >50K is printed.

CODE:

**import** pandas **as** pd  
**import** numpy **as** np  
  
**from** sklearn **import** preprocessing  
**from** sklearn.model\_selection **import** train\_test\_split, GridSearchCV  
**from** sklearn.naive\_bayes **import** GaussianNB  
**from** sklearn.tree **import** DecisionTreeClassifier  
**from** sklearn.neural\_network **import** MLPClassifier  
  
**import** seaborn **as** sns  
**import** matplotlib.pyplot **as** plt**import** warnings  
warnings.filterwarnings(**"ignore"**, category=DeprecationWarning)  
  
df = pd.read\_csv(**'Dataset\_2.csv'**)  
df.head()  
df.describe()  
df.info()  
  
corr = df.corr()  
print(corr.columns)  
sns.heatmap(corr)  
df.columns  
  
**for** i **in** df[**'Salary'**]:  
 print(**"-"**, i, **"-"**)  
 print(str.strip(i))  
 print(i.strip() **is "<=50K"**)  
 print(i.strip() == **"<==50K"**)  
 print((i == **"<=50K"**))  
 **break**df[**'Salary'**] = [0 **if** salary.strip() ==**"<=50K" else** 1 **if** salary.strip()==**">50K" else** salary **for** salary **in** df[**'Salary'**]]  
  
**for** column **in** df.columns:  
 df[column] = [value.strip() **if** type(value) == str **else** value **for** value **in** df[column]]  
 df.head()  
  
wc\_enc = preprocessing.LabelEncoder()  
X = df[**'WC'**]  
wc\_enc.fit(X.values)  
df[**'WC'**] = wc\_enc.transform(df[**'WC'**].values)  
  
el\_enc = preprocessing.LabelEncoder()  
X = df[**'EL'**]  
el\_enc.fit(X.values)  
df[**'EL'**] = el\_enc.transform(df[**'EL'**].values)  
  
ms\_enc = preprocessing.LabelEncoder()  
X = df[**'MS'**]  
ms\_enc.fit(X.values)  
df[**'MS'**] = ms\_enc.transform(df[**'MS'**].values)  
  
occ\_enc = preprocessing.LabelEncoder()  
X = df[**'Occ'**]  
occ\_enc.fit(X.values)  
df[**'Occ'**] = occ\_enc.transform(df[**'Occ'**].values)  
  
rs\_enc = preprocessing.LabelEncoder()  
X = df[**'RS'**]  
rs\_enc.fit(X.values)  
df[**'RS'**] = rs\_enc.transform(df[**'RS'**].values)  
  
gender\_enc = preprocessing.LabelEncoder()  
X = df[**'Gender'**]  
gender\_enc.fit(X.values)  
df[**'Gender'**] = gender\_enc.transform(df[**'Gender'**].values)  
df.head()  
  
columns\_to\_use = [**'Age'**, **'WC'**, **'EL'**, **'Year'**, **'MS'**, **'Occ'**, **'RS'**, **'Gender'**, **'CG'**, **'CL'**, **'Hours'**]  
  
X = df[columns\_to\_use]  
y = df[**'Salary'**]  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)  
print(X\_train.shape)  
print(X\_test.shape)  
  
  
**def** train\_with\_folds(clf):  
 *# split array in k(number of folds) sub arrays* X\_folds = np.array\_split(X\_train, 3)  
 y\_folds = np.array\_split(y\_train, 3)  
  
 scores = list()  
 models = list()  
 **for** k **in** range(3):  
 *# We use 'list' to copy, in order to 'pop' later on* X\_train\_fold = list(X\_folds)  
 *# pop out kth sub array for testing* X\_test\_fold = X\_train\_fold.pop(k)  
 *# concatenate remaining sub arrays for training* X\_train\_fold = np.concatenate(X\_train\_fold)  
  
 *# same process for y* y\_train\_fold = list(y\_folds)  
 y\_test\_fold = y\_train\_fold.pop(k)  
 y\_train\_fold = np.concatenate(y\_train\_fold)  
  
 clf = clf.fit(X\_train\_fold, y\_train\_fold)  
 scores.append(clf.score(X\_test\_fold, y\_test\_fold))  
 models.append(clf)  
  
 print(scores)  
  
*# using Gaussian Naive Bayes Method:*gnb = GaussianNB()  
gnb = gnb.fit(X\_train, y\_train)  
pred = gnb.predict(X\_test)  
print(**"mean accuracy for Gaussian naive bayes"**,gnb.score(X\_test, y\_test))  
train\_with\_folds(clf=GaussianNB())  
  
*# using Decision Tree Classifier:*clf = DecisionTreeClassifier(max\_depth=10)  
clf = clf.fit(X\_train, y\_train)  
print(**"mean accuracy for Decision Tree Classifier "**, clf.score(X\_test, y\_test))  
clf = DecisionTreeClassifier(max\_depth=10)  
train\_with\_folds(clf)  
  
*# using MLP classifier:*clf = MLPClassifier(solver=**'adam'**, activation=**'tanh'**, alpha=1e-5, hidden\_layer\_sizes=(15, 5), random\_state=43)  
clf = clf.fit(X\_train, y\_train)  
print(**"mean accuracy for MLP"**, clf.score(X\_test, y\_test))  
clf = MLPClassifier(solver=**'adam'**, activation=**'tanh'**, alpha=1e-5, hidden\_layer\_sizes=(9, 2), random\_state=43)  
train\_with\_folds(clf)  
  
*# printing the profile of all the persons whose salary is likely >50K*print(type(pred))  
pred\_index = 0  
**for** row **in** np.nditer(pred):  
 **if** row == 1:  
 POI = X\_test.iloc[pred\_index]  
 print(**"required profile is "**,gender\_enc.inverse\_transform([POI[**'Gender'**]])[0], **" "**, POI[**'Age'**] ,**" "**, \  
 occ\_enc.inverse\_transform([POI[**'Occ'**]])[0], **" "**,rs\_enc.inverse\_transform([POI[**'RS'**]])[0])  
  
 pred\_index += 1

Screenshots:

