|  |  |
| --- | --- |
|  | **DOKUZ EYLÜL UNIVERSITY**  **ENGINEERING FACULTY**  **DEPT. OF COMPUTER ENGINEERING** |

# CME 4416 Introduction to Data Mining

# Final Project Report

**2019-2020 SPRING**

**2016510086**

**Mustafa ÖZSARAÇ**

# 1-Introduction

In this progress report, I will explain my process for the final project. This report will contain explanation of what I have done, description of my data set and what techniques I applied in order to prepare the dataset. Also, the results for each algorithm to apply for this dataset and problem. We will do a “**classification**” as topic for this project. My aim for this final project will be to create a system which can accurately classify this dataset into the right categories. We will use Adult – Income dataset as I talk about it in section 2.

# 2-Information About Dataset

This dataset is called Adult Census Income and contains information of adults, and includes information such as age, education, race, sex, work hours per week, native country and income among other things. Target feature is “income”. We will be predicting the class of the people based on their salary, either making <=50k or >50k per year.

**COLUMN DESCRIPTIONS:**

* **Age** 🡪 Age of that person.
* **Workclass** 🡪 It’s about where that person work. Such as government, self-employed etc.
* **Fnlwgt** 🡪 This feature represents final weight, which is the number of units in the target population that the responding unit represents.
* **Education** 🡪 Education level of that person.
* **Education\_num** 🡪 Number of education.
* **Marital\_status** 🡪 Represent marital status of that person. It can be “Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse”.
* **Occupation** 🡪 Represents job of that person. Values are “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** 🡪 Such as “Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried”.
* **Race** 🡪 White, Asian-Pac-Islander, Amer-Indian-Eskimo, Black, other
* **Sex** 🡪 Female, Male.
* **Capital Gain** 🡪 Represents a rise in the value of a capital asset (investment or real estate) that gives it a higher worth than the purchase price.
* **Capital Loss** 🡪 Represents a decrease in the value of a capital asset (investment or real estate) that gives it a higher worth than the purchase price.
* **Hours per week** 🡪 Working hours per week(continuous)
* **Native Country** 🡪 Values are : “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”.

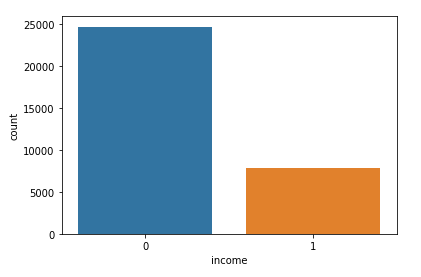
**TARGET FEATURE:**

* **Income:** Income of that person, whether it’s bigger than 50K or less.

Now, let’s do some **EDA** **(exploratory data analysis)** to know the data set a little bit better.

**Q1: How is the distribution in income?**

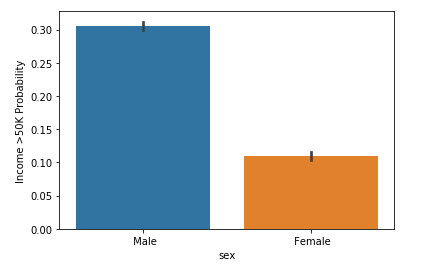




As you can see almost 25K people from this data set earns less than 50K.

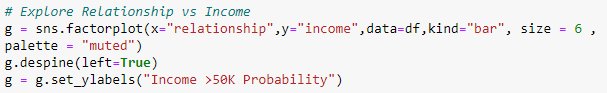
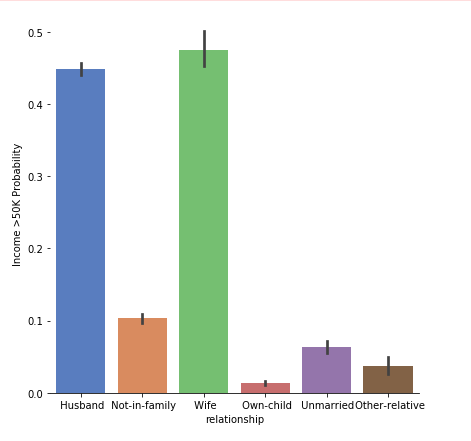
**Q2: How is the distribution of income in case of sex?**





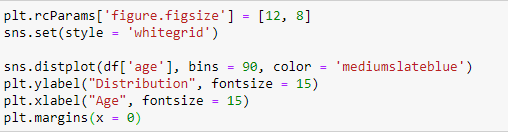
As you can see from the graphs, male has more probability to earn more than 50K.

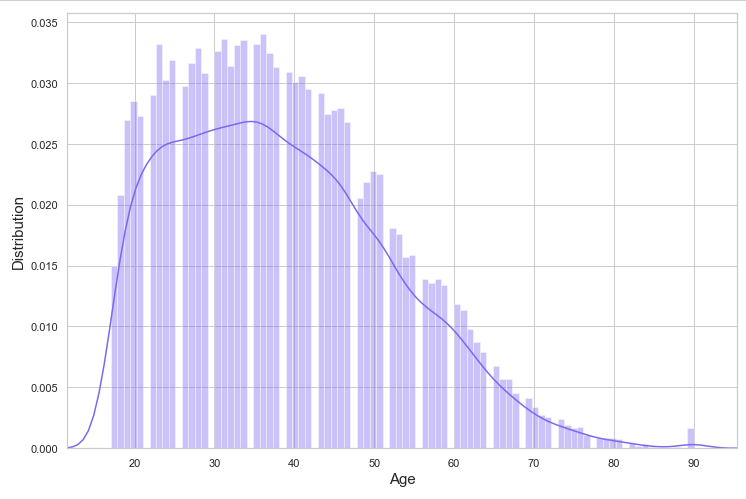
**Q3: How is the distribution of income in case of relationship?**



As you can see from the table, marital status effects income. Husbands and Wives are tending to earn more than 50K.

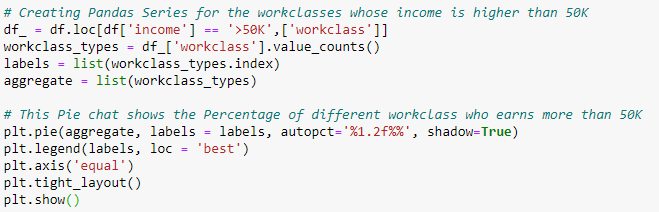
**Q4: How is the distribution of the distribution of age of people?**

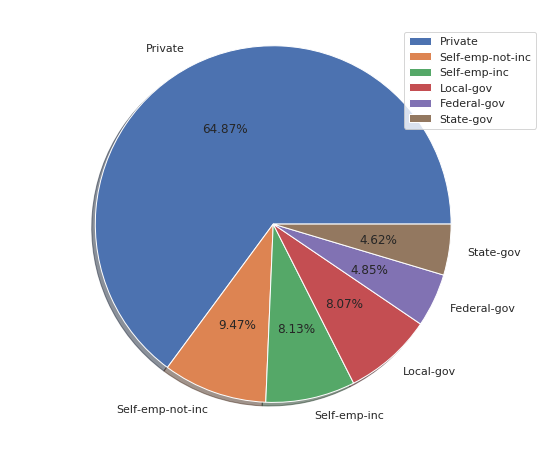




As you can see from the graph, youngest person is 17, and oldest is 90.

**Q5: How is the distribution of the jobs of people who earn more than 50K?**

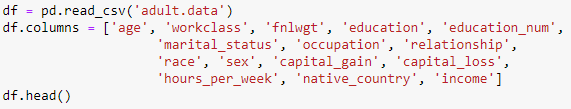




As you can see most of the people who make more than 50K, is working privately.

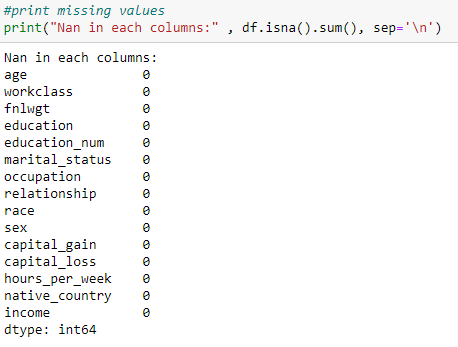
# 3-Data Preparation Techniques

I used **Python** with **Jupyter Notebook** and “**pandas**” library to load this data as data frame. To read the data, I used the code shown below:



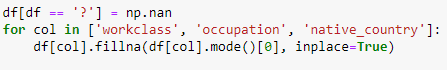
First I read the data file with pd.read\_csv command, then I add column names because this data file did not have them.

Then, I print the list of NA values in data frame:

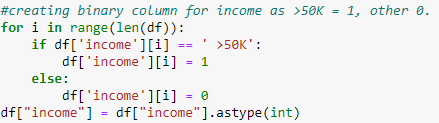


As you can see there’re no missing values. Or it seems that way.

Later on I found out that there some missing values but those are represented as “?” not NaN. Workclass, occupation and native country has some “?” values. So replaced them with the code shown below:



Then, I though making income binary would be better. So if it’s bigger than 50K it should be 1, and if it’s lower 0.

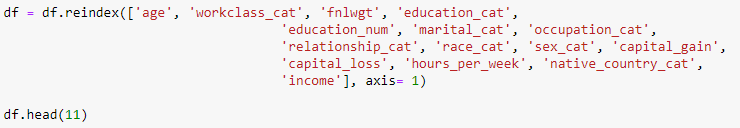


Now that this is done, I need to encode categorical data. I used LabelEncoder to perform such operation. Code shown below:

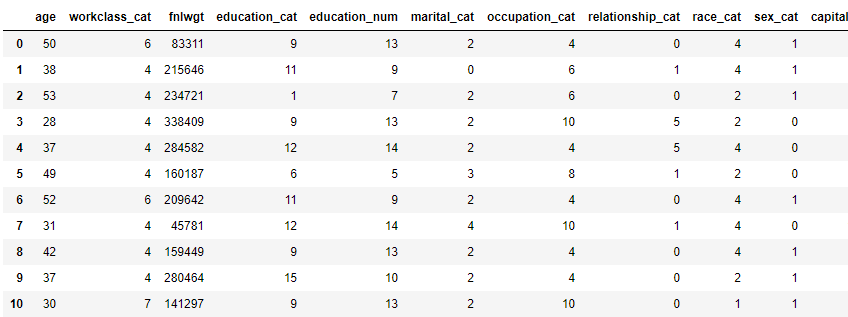


In here I create new columns with \_cat name on it. These are the ones that transformed. After applying this to every categorical data, I removed the old columns which are useless now.

Then indexes were wrong because I did a lot of adding and removing columns. So, I reindexed all these columns:

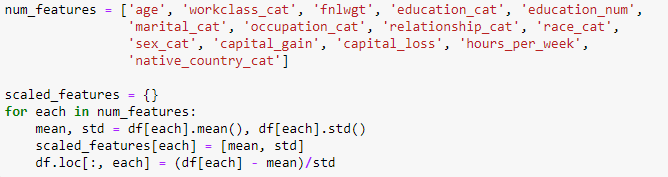


Now, it looks like this:



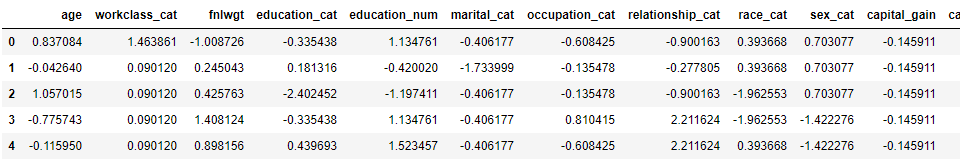
It’s all numerical.

Now, I need to scale these columns because numbers weight can vary.



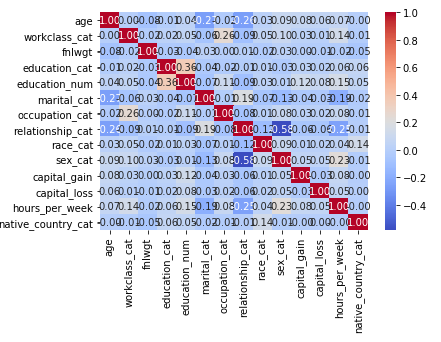
After that, all features are scaled now.

Our data frame looks like this now:



After that I wanted to check on correlation map. Code as follows:





Now, I can move on to Model Implementation part.

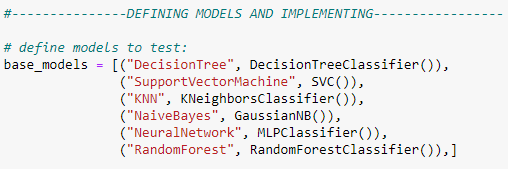
# 4- Model Implementation & Results

Since data preprocessing steps are done, we can move on to model implementation. In order to do that we must separate features first.

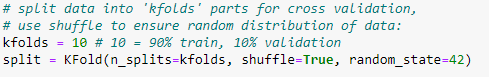


So, X is all the features except target feature which is “income” and y is just “income”.

After that, I defined the models:

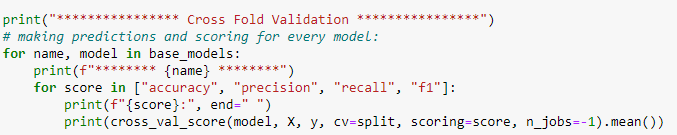


And then, to use cross fold validation I created necessary variables:



I choose 10 for folding, shuffle = True to ensure the random distribution of data. Random\_state = 42 is for ensuring same results will occur for the same splitting.

After that, I can use cross fold validation to test algorithms, and print the results.



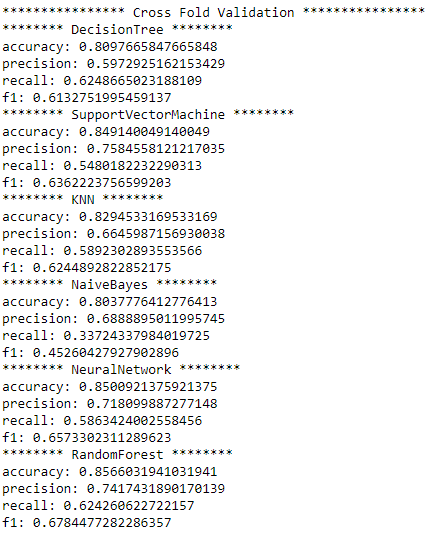
This codes here do these as listed below:

• Firstly, gets the model from base\_models array and prints the name of model that will be used.

• Then, it chooses scoring metric from the array of accuracy, recall, precision and f1-score.

• Then, it will get the score of wanted metric and wanted model in order. Finally, it will print these results to the console.

The output of the results is as shown below:



As you can see from the scores, **Random Forest** performs best for cross fold validation with the **highest accuracy and f1-score**.

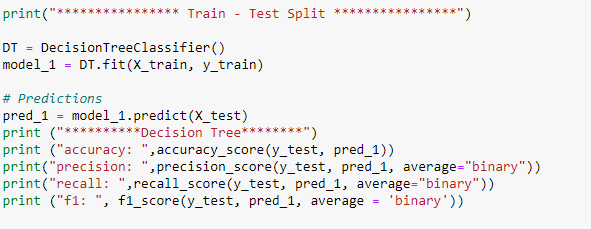
Now, I need to test the algorithms with train – test split. So, in order to do that I need to split the data first.



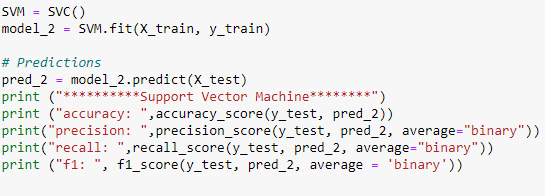
I split the data to %90 training %10 testing as wanted. Also imported some necessary libraries.

Now I can test each algorithm with train – test split.

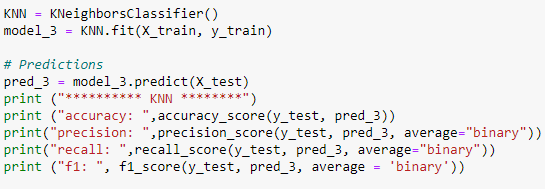
Codes are as shown below:



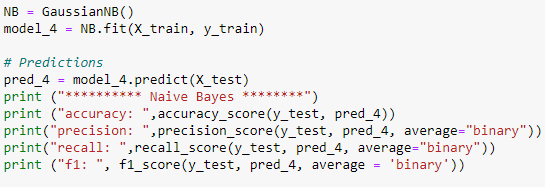
Decision Tree Classifer with train – test split.



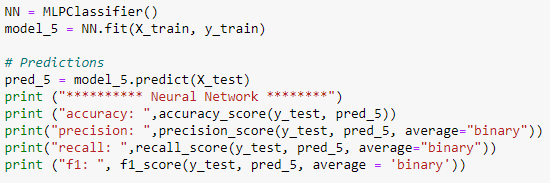
Support Vector Machine with train – test split.



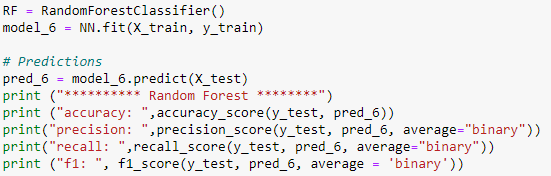
KNNClassifer with train - test split.



Naïve Bayes with Gaussian formula.

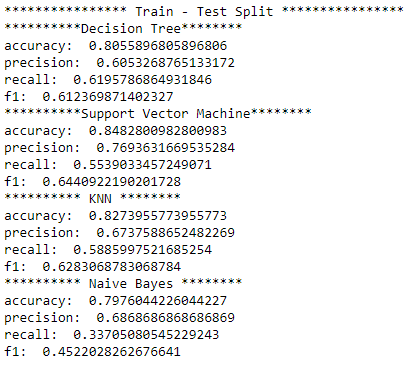


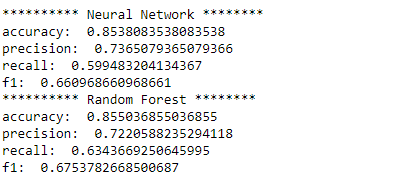
Neural Network with train – test split.



And finally, Random forest with train – test split.

Results are as shown below:

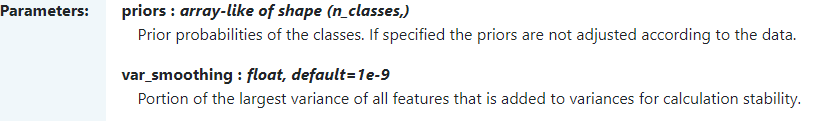




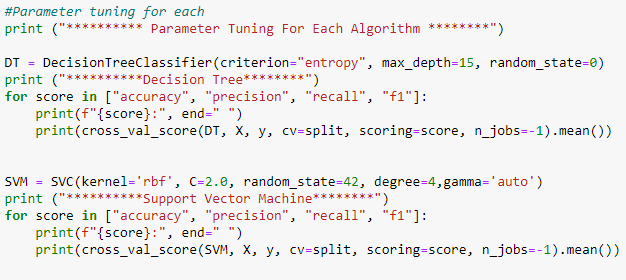
As you can see from the results, with the train – test split Random Forest performs best once again. **It has highest accuracy and f1 score** of all.

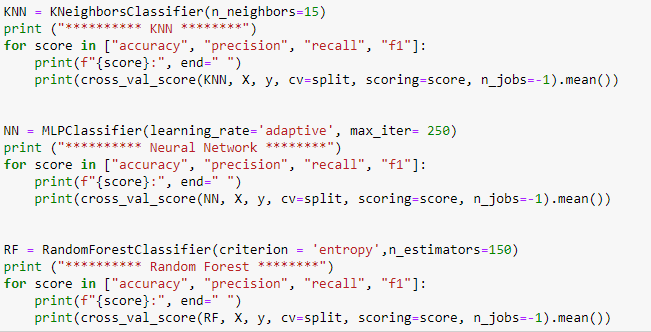
# 5-Parameter Tuning for Each Algorithm

Now, I can try to improve accuracy and f1-score values by changing some parameters for each algorithm except Naïve Bayes. It’s because GauissianNB() has only two parameters and these are the things that it’s better stay in default. Parameters are as follows:

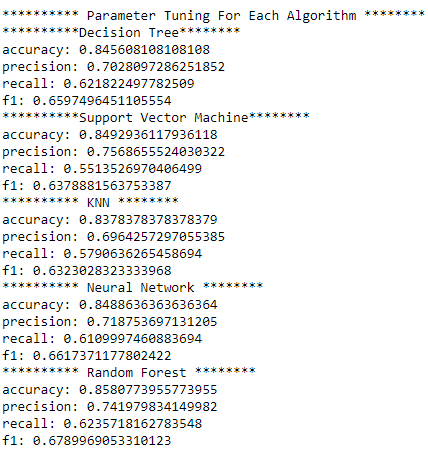


For the other five algorithms, these are the best that I found are as shown below:





And results are as follows:



Once again, **Random Forest performs best with %85 accuracy and %67 f1-score**. But **Decision Tree improves most** because it’s accuracy jumped from %80 to %84.5 so there’s a %4.5 increase on it.

# 6-Conclusion

Table of algorithms with **Cross Fold Validation**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1-score |
| Decision Tree | 0.809 | 0.597 | 0.624 | 0.613 |
| SVM | 0.849 | 0.758 | 0.548 | 0.636 |
| KNN | 0.829 | 0.664 | 0.589 | 0.624 |
| Naïve Bayes | 0.803 | 0.688 | 0.337 | 0.452 |
| Neural Network | 0.850 | 0.718 | 0.586 | 0.657 |
| Random Forest | 0.856 | 0.741 | 0.624 | 0.678 |

Table of algorithms with **Train – Test Split (%90-10)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1-score |
| Decision Tree | 0.805 | 0.605 | 0.619 | 0.612 |
| SVM | 0.848 | 0.769 | 0.553 | 0.644 |
| KNN | 0.827 | 0.673 | 0.588 | 0.628 |
| Naïve Bayes | 0.797 | 0.686 | 0.337 | 0.452 |
| Neural Network | 0.853 | 0.736 | 0.599 | 0.660 |
| Random Forest | 0.855 | 0.722 | 0.634 | 0.675 |

Table of algorithms with cross fold and **Parameter Tuning**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1-score |
| Decision Tree | 0.845 | 0.702 | 0.621 | 0.659 |
| SVM | 0.849 | 0.756 | 0.551 | 0.637 |
| KNN | 0.837 | 0.696 | 0.579 | 0.632 |
| Neural Network | 0.848 | 0.718 | 0.610 | 0.661 |
| Random Forest | 0.858 | 0.741 | 0.623 | 0.678 |

As you can see from the three tables above, Random Forest performs best among these six algorithms. Random Forest provides **best accuracy, recall and f1-score** and second highest precision. Thus, Random Forest is the best algorithm for this dataset. Other than Random Forest, my second choice would be **Neural Network** with second at accuracy, recall, and F1-score.