Assignment 1: Classification

John Christian Castillo - 20175247

April 16, 2023

---

1 Introduction

All the files for this project is uploaded to github and can be found here.

2 Preprocessing

Before we can run any algorithm to our data, it is ideal for said data to be in the ideal format. This is where preprocessing comes into play.

2.1 Parsing

The first step in preprocessing is reading the data, which in our case is an .xlsx file. This was done with the help of the library **pandas** and its built in function **read_excel()**. I made the choice to throw out the first column **RowID** as I believe this to be not of any use.

Existing Customers:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba
...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States
32557	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States
32558	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States
32559	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States
32560	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States

32561 rows × 13 columns

Figure 1: Existing Customers

2.2 Data conversion

After reading the data it can be observed that their types is not homogeneous. To achieve this, I made use of **preprocessing** which is included in the library **sklearn**. Within **preprocessing** exists the class **OrdinalEncoder()** which contains the function **fit()** and **transform()**. Which ultimately allowed for the data to become homogeneous. This can be observed in the figure 2, which is the Transformed Potential Customers.

Transformed Potential Customers:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	8.0	3.0	1.0	6.0	4.0	6.0	3.0	2.0	1.0	0.0	0.0	39.0	37.0
1	21.0	3.0	11.0	8.0	2.0	4.0	0.0	4.0	1.0	0.0	0.0	49.0	37.0
2	11.0	1.0	7.0	11.0	2.0	10.0	0.0	4.0	1.0	0.0	0.0	39.0	37.0
3	27.0	3.0	15.0	9.0	2.0	6.0	0.0	2.0	1.0	90.0	0.0	39.0	37.0
4	1.0	NaN	15.0	9.0	4.0	NaN	3.0	4.0	0.0	0.0	0.0	29.0	37.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
16276	22.0	3.0	9.0	12.0	0.0	9.0	1.0	4.0	0.0	0.0	0.0	35.0	37.0
16277	47.0	NaN	11.0	8.0	6.0	NaN	2.0	2.0	1.0	0.0	0.0	39.0	37.0
16278	21.0	3.0	9.0	12.0	2.0	9.0	0.0	4.0	1.0	0.0	0.0	49.0	37.0
16279	27.0	3.0	9.0	12.0	0.0	0.0	3.0	1.0	1.0	75.0	0.0	39.0	37.0
16280	18.0	4.0	9.0	12.0	2.0	3.0	0.0	4.0	1.0	0.0	0.0	59.0	37.0

Figure 2: Transformed Potential Customers

2.3 Multivariate Imputation

Analyzing figure 2 however, it can be seen that on row 16277, column 1 has a cell that is not filled which results for it to show "NaN". This brings us to the next step which is to fill these empty cells. Luckily there exists a class within **sklearn.impute** called **IterativeImputer** which can perform a multivariate imputation to our dataset. Applying this results into figure 3 where it can be observed that the problem from the above mentioned (row, column) combination is gone and is replaced by an actual value.

Transformed Potential Customers:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	8.0	3.000000	1.0	6.0	4.0	6.000000	3.0	2.0	1.0	0.0	0.0	39.0	37.0
1	21.0	3.000000	11.0	8.0	2.0	4.000000	0.0	4.0	1.0	0.0	0.0	49.0	37.0
2	11.0	1.000000	7.0	11.0	2.0	10.000000	0.0	4.0	1.0	0.0	0.0	39.0	37.0
3	27.0	3.000000	15.0	9.0	2.0	6.000000	0.0	2.0	1.0	90.0	0.0	39.0	37.0
4	1.0	2.974661	15.0	9.0	4.0	5.575642	3.0	4.0	0.0	0.0	0.0	29.0	37.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
16276	22.0	3.000000	9.0	12.0	0.0	9.000000	1.0	4.0	0.0	0.0	0.0	35.0	37.0
16277	47.0	3.126844	11.0	8.0	6.0	5.796052	2.0	2.0	1.0	0.0	0.0	39.0	37.0
16278	21.0	3.000000	9.0	12.0	2.0	9.000000	0.0	4.0	1.0	0.0	0.0	49.0	37.0
16279	27.0	3.000000	9.0	12.0	0.0	0.000000	3.0	1.0	1.0	75.0	0.0	39.0	37.0
16280	18.0	4.000000	9.0	12.0	2.0	3.000000	0.0	4.0	1.0	0.0	0.0	59.0	37.0

16281 rows × 13 columns

Figure 3: Transformed Potential Customers(Imputed)

3 Classification

After the above steps are done, the data can now be processed. This leads us to the question: "Which algorithms do we want to use?". For this, **scikit learn** actually provided us with a handy figure shown in figure 4. From the node labeled "START" a path can be made to the area labeled "classification". This is because:

1. There are > 50 samples
2. The goal is to predict a category and not a quantity ($\leq$ 50k or not)
3. The data is labeled

Within "classification" the path to "Linear SVC" is then taken as there are less than 100k samples. The methods that follows this branch has been used to attain a model that is as accurate as possible. These methods gave the following results:

1. Linear SVC: 0.824
2. Naïve Bayes: 0.812
3. K Nearest Neighbor: 0.830
4. SVC: 0.815
5. Decision Tree: 0.844
6. Ensemble Classifiers:
   (a) Bagging (Decision Tree): 0.857
   (b) Bagging (KNN): 0.830
   (c) Bagging (Naïve Bayes): 0.812
   (d) Random Forest: 0.863

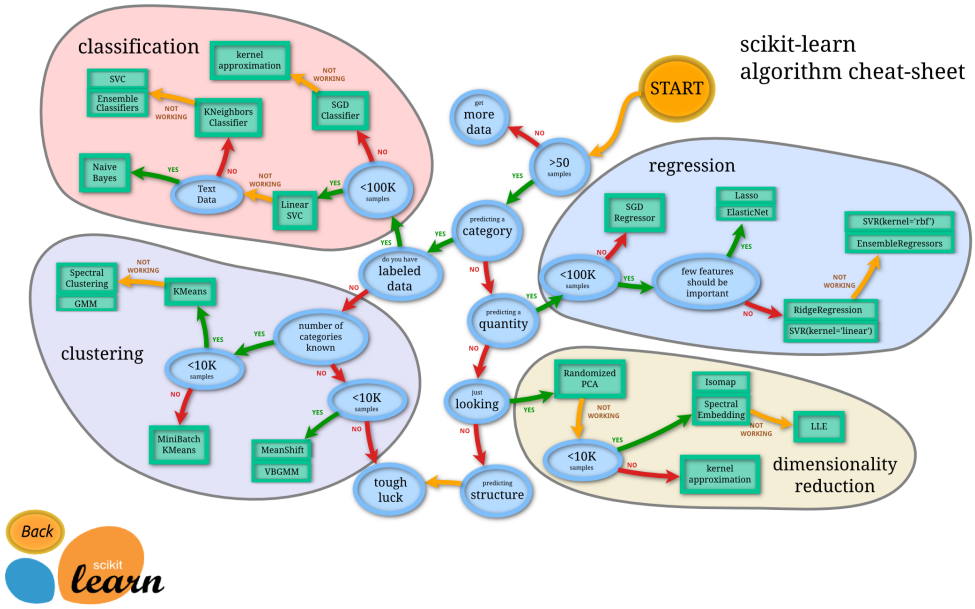

Figure 4: Algorithm cheat-sheet

4 Prediction

This leads to the conclusion to use the Random Forest classifier to predict which potential customers belong to the $\leq$ 50k group and which are not. For this, the function **predict˙proba()** is used as this gives us an idea of how certain the classifier was in appointing a label to a specific entry.

4.1 High earning customers

To get an idea of how many high earning customers there actually are, I decided to only make use of the data for which the model is at least 70% sure that something it labeled "High earning" is actually high earning. More specifically, to calculate the revenue due to high earning customers, the below code is used:

```
high_earning_revenue = len(high_earning) * 0.1 * 980 - len(high_earning)
                       * 0.1 * 10
```

In which the average profit is multiplied by the expected (10%) amount of high earning potential customers that will actually buy the promotion. After which the cost of shipping is subtracted for every package shipped.

4.2 Low earning cost

Since it is not ideal to send all low income customers the special offer, I decided to only choose those of which that may be incorrectly labeled. These are individuals for which the classifier is only roughly 50% sure of its labeling. Taking this risk for an individual may cost us 320 euros. But on the other hand, if the low income customer is actually mislabeled, it can net us 970 euros in profit. Knowing this I averaged the cost/profit that can be obtained by taking this risk within the accuracy range of 45% to 50%. Putting this all together results in the code below:

```
low_earning_cost = ((len(low_earning) * 0.05 * 310)
                    - (len(low_earning) * 0.1 * 980))/len(low_earning)
                    + len(low_earning) * 0.1 * 10
```

5 Conclusion

Adding the calculated "high_earning_revenue" and the "low_earning_cost", a total revenue of "144349.5" is obtained. Furthermore, the selected individuals is stored in the file called "selected.txt".