MACHINE LEARNING PROJECT

Yagmur Aktas
Condorcet university center
University of Burgundy
Le-Creusot, France
aktas.yagmur@gmail.com

Osarumen Osazuwa
Condorcet university center
University of Burgundy
Le-Creusot, France
sololambert20@gmail.com

Divine Bakala
Condorcet university center
University of Burgundy
Le-Creusot, France
divinebakala@yahoo.fr

April 28, 2021

ABSTRACT

Measures of personal characteristics, home ground, educational history and type of primary school attended were obtained for a representative sample of 500 eleven-year old children attending primary school in Ireland. Our goal is to understand the relation between socio-economic status, school attendance record, type of primary school attended, verbal reasoning ability, personal characteristics like sex and the certificate level.

1 Introduction

The goal of this study is to understand the relations between the data features and train a model best fitting to have correct results on our classification problem. The classification problem is based on prediction of the certificates taken or not taken by students having different values on 5 category. Since our target class contains 2 different values, it is a 'binary classification' problem and we used Logistic Regression to create our model.

As the second part, we compared various Non Linear Regressors to see the different performances

2 Data Inspection of Irish dataset

Our data contains
Four nominal category: Sex (2 unique values and 0 missing value), Educational level(10 unique values and 6 missing value), Type school(3 unique values and 0 missing value), Leaving Certificate (2 unique values and 0 missing value)
Two numerical: DVRT (68 unique values and 0 missing value) and Prestige score (28 unique values and 26 missing value)

In figure 1, we see a brief resume of our dataset.

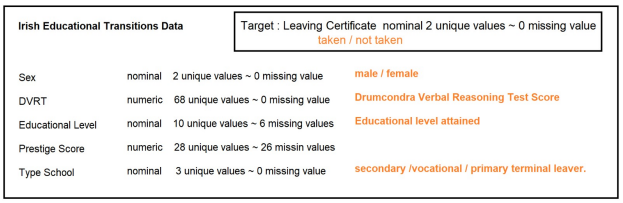

Figure 1:

In figure 2, we see the first 5 lines of our data using .head() command.

	Sex	DVRT	Educational_level	Leaving_Certificate	Prestige_score	Type_school
0	male	113	Junior_cycle_incomplete-secondary_school	not_taken	28.0	secondary
1	male	101	Primary_terminal_leaver	not_taken	28.0	primary_terminal_leaver
2	male	110	Senior_cycle_terminal_leaver-secondary_school	taken	69.0	secondary
3	male	121	Junior_cycle_terminal_leaver-secondary_school	not_taken	57.0	secondary
4	male	82	Junior_cycle_terminal_leaver-vocational_school	not_taken	18.0	vocational

Figure 2: Visualizing the first 5 lines of our Data

After a first look we used Panda Correlation method to inspect the correlation between our features.

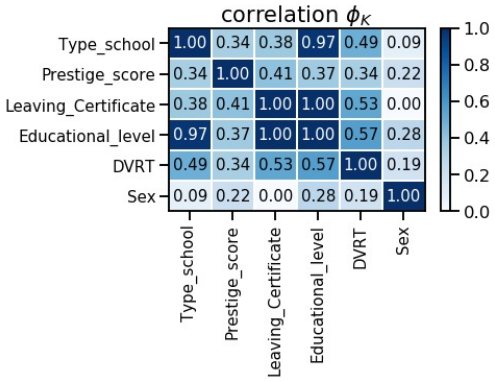

Figure 3: Correlation Matrix

We also check our numerical columns and we saw that the range of DVRT and Prestige Score were different which means we will have to use StandardScalar as Preprocessor.

	DVRT	Prestige_score
count	500.000000	474.000000
mean	100.152000	38.934599
std	15.456348	15.333707
min	65.000000	18.000000
25%	90.000000	28.000000
50%	101.000000	37.000000
75%	111.000000	46.000000
max	140.000000	75.000000

Figure 4:

2.1 Visualizing the Nominal Data using OneHotEncoder

We visualized the categorical data before using a OneHotEncoder in model pipeline to see the change. Using OneHotEncoder each unique value in a category becomes a column; the encoding returned, for each sample, a 1 to specify which category its belongs to.

2.2 The features and Distribution of their Values

We see that there are classes having just 1 sample, especially in DVRT and Prestige Score features. So we should use handle known = 'ignore' to prevent that these samples go to the test dataset and never seen by the model in training stage.

Sex One Hot Encoded

	Sex_female	Sex_male
0	0.0	1.0
1	0.0	1.0
2	0.0	1.0
3	0.0	1.0
4	0.0	1.0
..	...	...
495	0.0	1.0
496	0.0	1.0
497	0.0	1.0
498	1.0	0.0
499	1.0	0.0

Type_school One Hot Encoded

	Type_school_primary_terminal_leaver	...	Type_school_vocational
0	1.0	...	0.0
1	0.0	...	0.0
2	0.0	...	0.0
3	0.0	...	1.0

Educational_level One Hot Encoded

	Educational_level_Junior_cycle_terminal_leaver-secondary_school	...	Educational_level_Senior_cycle_terminal_leaver-secondary_school
0	0.0	...	0.0
1	0.0	...	1.0
2	1.0	...	0.0
3	0.0	...	0.0

Figure 5: Visualizing the Encoded version of our Nominal Categorical Columns

Sex	
male	250
female	250

Educational_level	
Senior_cycle_terminal_leaver-secondary_school	158
Junior_cycle_terminal_leaver-vocational_school	68
Junior_cycle_terminal_leaver-secondary_school	65
3rd_level_complete	57
Junior_cycle_incomplete-vocational_school	50
Primary_terminal_leaver	37
Junior_cycle_incomplete-secondary_school	30
Senior_cycle_incomplete-vocational_school	13
Senior_cycle_incomplete-secondary_school	9
3rd_level_incomplete	7

Type_school	
secondary	325
vocational	138
primary_terminal_leaver	37

DVRT	
70	30
104	19
90	18
100	17
103	17
..	..
137	1
130	1
131	1
132	1
65	1

Prestige_score	
18.0	91
37.0	89
43.0	46
28.0	39
46.0	6
36.0	5
65.0	5
42.0	4
66.0	3
30.0	3
47.0	2
48.0	1
53.0	1
64.0	1

dtype: int64

Figure 6: Features and Distribution of their Values

3 Creating and Improving the Model

3.1 Creating the First Model

Since our dataset contains missing values, we used Simple Imputer to replace these missing values with most frequent value. Then we replaced nominal target data values to numeric values : 0 (not taken) and 1 (taken)

To create our model pipeline we used the following preprocessors for both the categorical and numerical:

1.categoricalpreprocessor = OneHotEncoder(handle unknown="ignore")
2.numericalpreprocessor = StandardScaler()
3.preprocessor = ColumnTransformer([('one-hot-encoder', categoricalpreprocessor, categoricalcolumns), ('standardscaler', numericalpreprocessor,numericalcolumns),])

We then created our model pipeline:
model = make pipeline(preprocessor, LogisticRegression(penalty="l2")

First result of our model:
fit time =[0.12269163 0.12235808 0.12254596 0.12453055 0.12268996]
score time =[0.013937 0.01443076 0.01444435 0.012712 0.01592565]
train score =[0.99333333 0.99333333 0.99 0.99 0.99]
test score =[0.98 0.975 0.985 0.985 0.985]
The accuracy is: 0.982 +- 0.004

3.2 Examining the Validation and Learning Curve

We examined three different validation curves based on these parameters of our model:

*Max iter Parameter Range = [50, 100, 250, 500, 1000, 2000]

*C Regularization Parameter range = [0.2, 0.5, 1, 5, 10, 20]

*Weights = np.linspace(0.05, 0.95, 6); Parameter Range = [x for x in weights]

While the Learning Curve depends on the sample sizes, we examined the learning curves with this range of dataset size : [50, 100, 150, 200,250, 300]

Having 500 samples in our dataset, for the first trial, we split the data 40 percent for test and 60 for train. For the second trial we split them 20 percent for test and 80 percent for train. We see below the validation and learning curves having these 2 different percentage of data splitting.

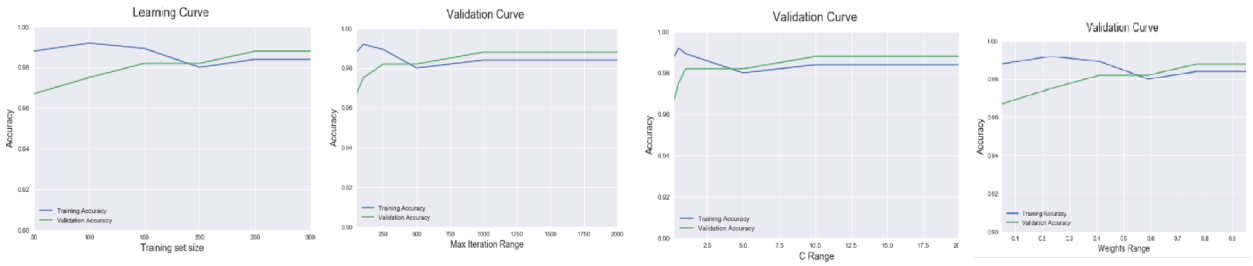

Figure 7: First Validation and Learning Curve

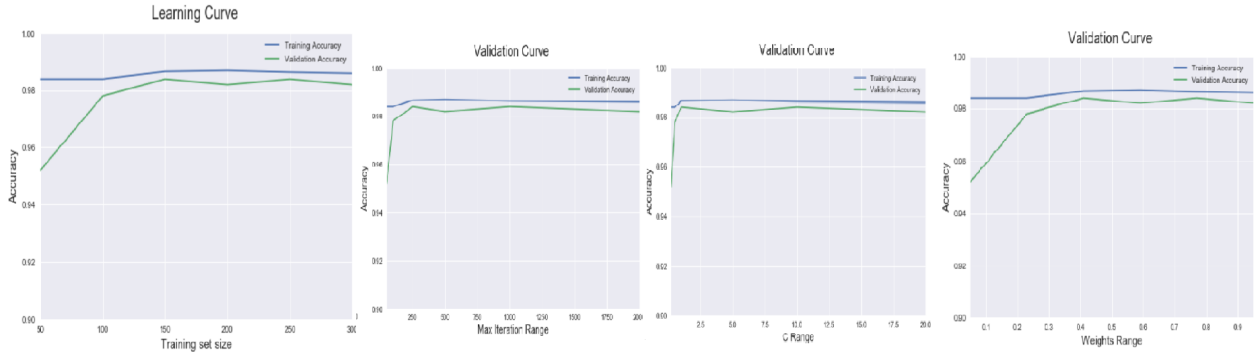

Figure 8: Second Validation and Learning Curve

We noticed the train score get lower when the validation score gets higher after a while for the first trial. So it is not "overfitting", and not really "underfitting" too, because the accuracy was not so bad, but the performance is not good. Second percentage gives a better result than first one but it stops to increase the accuracy for both validation and train score. For that reason, we chose the second one to see the result with fine tuned parameters.

3.3 Chosen Model

Max iter Parameter = 250, C Regularization Parameter= 1, class weight = 0.25
Result of the Model with Fine Tuned HyperParameters:
train score: [0.99 0.9925 0.985 0.9925 0.985]
test score :[0.98 0.97 0.99 0.97 1.] The accuracy is: 0.982 +- 0.012
As the last train for that model, we trained the fine tuned model with only more correlated parameters to see the

```
Result of the Model with Fine Tuned HyperParameters and with only high Correlated
Features:
train_score
[0.98333333 0.98333333 0.98       0.99       0.99333333]
test score
[0.995 0.995 0.995 0.975 0.98 ]
The accuracy is: 0.988 +- 0.009
```

Figure 9: Second train for the fine tuned model with more related features

difference and as a result we obtain a better accuracy at 0.006 as shown in Figure 9.

As the last step, we will compare this model's performance with Non Linear SVM and Decision Tree Classifiers with different depths.

4 Comparing our Models with Non Linear and Decision Trees Models

We see the Non Linear model has a better accuracy at + 0.02 After using 3 different decision trees with max depths 3,5 and 10, we see none of them have a better accuracy then our model and it gets lower with higher depth.

```
Non Linear SVM
[0.99   0.9875 0.985  0.9875 0.9875]
test score
[0.98 0.99 1.   0.99 0.99]
The accuracy is: 0.990 +- 0.006
```

```
Decision Tree Classifier Max Depth = 3
train score
[0.985  0.9875 0.985  0.9925 0.9925]
test score
[1.   0.99 0.97 0.97 0.98]
The accuracy is: 0.982 +- 0.012
```

```
Decision Tree Classifier Max Depth = 5
train score
[0.99   0.99   0.995  0.9875 0.995 ]
test score
[0.98 0.99 0.98 0.99 0.98]
The accuracy is: 0.984 +- 0.005
```

```
Decision Tree Classifier Max Depth = 10
train score
[1. 1. 1. 1. 1.]
test score
[0.96 0.99 0.98 0.99 0.98]
The accuracy is: 0.980 +- 0.011
```

Figure 10: The Results of Nonlinear and Decision Trees Models

5 Conclusion

Even with fine tuning or using only the correlated data to train the model, we see Non Linear SVM gives a better accuracy. As a result, having a Non Linear model for our dataset and improving it will be more useful for this classification problem.