University of Essex

School of Computer Science and Electronic Engineering (CSEE)

Subject: Comparative Study Report (Coursework Part 2)

Module: CE802 – Machine Learning and Data Mining

Author:

Shahrukh
(sv20035@essex.ac.uk)

04-August-2021

Contents

Part 2: Tosco & Spency

Introduction

This report is based on customer data of Tosco & Spency (a famous supermarket chain) seeking for some hidden parameters of their customers such as predicting the class of a customer visiting the supermarket from one of the two below:

1. If the customer is prone to buy few expensive products (Target: True or 1)
2. Or buys many cheap products (Target: False or 0)

In this classification problem, I am given the historical data of past customers with 15 different features and ground truth class variable, I first mine and preprocess the data to inspect for any errors and to transform the data into an optimized and acceptable format for implementing ML techniques.

Data Preprocessing

We begin by loading the data using Pandas library.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	Class
0	-1	45	-7.97	0.54	3	0.50	-113.34	20.30	9.99	20.50	-5.52	3.60	-466.22	1.88	-7.49	True
1	1	27	-7.02	1.08	3	0.85	-47.34	20.00	2.88	19.40	-6.49	1.22	-470.22	0.57	NaN	True
2	-2	0	-8.82	0.56	3	0.45	-152.34	19.62	9.78	20.74	-5.21	2.08	-534.22	5.62	-5.74	False
3	-14	855	-3.23	12.00	30	7.45	-341.34	34.76	-10.14	14.38	-4.79	-2.52	-846.22	-4.17	NaN	True
4	-1	39	-8.12	2.88	3	0.76	-53.34	19.08	6.48	22.58	-7.52	1.24	-512.22	2.17	NaN	False

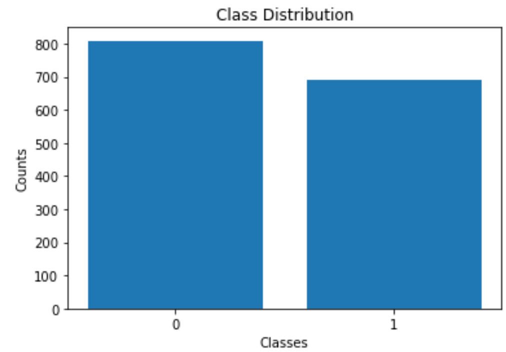
While inspecting the data, we see that all the features are in numeric format already, except the target variable "**Class**", which is in bool format. Although python cast bool variables implicitly to integers (True=1, False=0) when processing but it is a good practice to have explicit integer representation. So I have converted "Class" variable to integer.

Another thing that we notice from the picture on the left, that the feature "F15" has only 750 non-null values. You can find further about the null values percentages for each feature in the notebook.

```
Data columns (total 16 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   F1      1500 non-null   int64
 1   F2      1500 non-null   int64
 2   F3      1500 non-null   float64
 3   F4      1500 non-null   float64
 4   F5      1500 non-null   int64
 5   F6      1500 non-null   float64
 6   F7      1500 non-null   float64
 7   F8      1500 non-null   float64
 8   F9      1500 non-null   float64
 9   F10     1500 non-null   float64
 10  F11     1500 non-null   float64
 11  F12     1500 non-null   float64
 12  F13     1500 non-null   float64
 13  F14     1500 non-null   float64
 14  F15     750 non-null    float64
 15  Class   1500 non-null   bool
dtypes: bool(1), float64(12), int64(3)
```

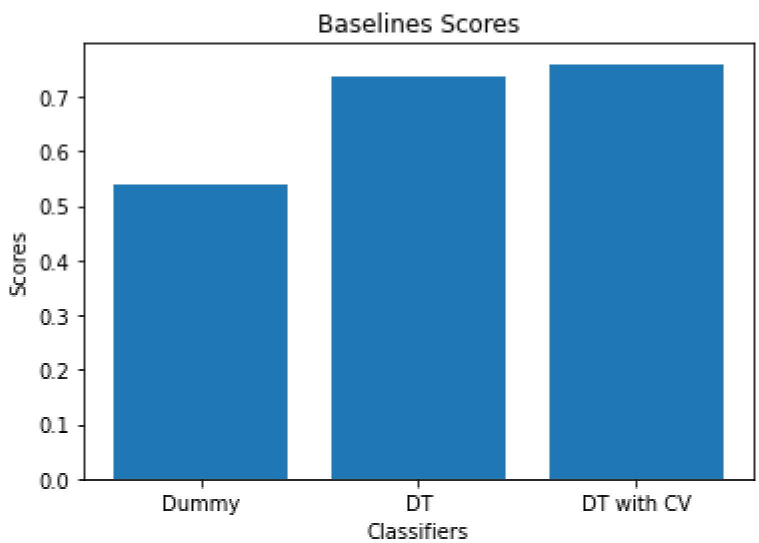
Evaluation Metric

I am using the simplest evaluation metric that is Accuracy Score. F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Whereas, it is a balanced class distribution problem. Hence, we are more concerned about the true positives and true negatives together whereas, F1 is a harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases which is not necessarily needed here.



Baseline Approach

As a baseline approach, I first have simply dropped the feature with null values and without performing any data transformations to tackle biasness and variances of the ML models. I have tried a DummyClassifier and DecisionTreeClassifier (with and without Cross-Validation) to achieve baseline accuracy scores.

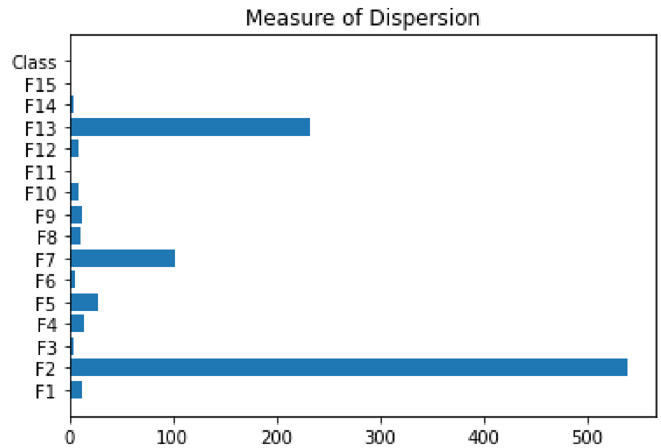


It is good to see that DecisionTreeClassifiers (with or without CV) are performing better than ordinary or common anticipation of DummyClassifier which predicts the most frequent/modal class from the fitted data.

Comparative Study

Now I am going to tune the data to remove misleading and misdirecting parameters leading the model to become biased or have large variances. First, I have used IQR (Interquartile Range) or H-Spread, a statistical dispersion, to find out the outliers. This method is quite robust to finding the outliers.

```
<function matplotlib.pyplot.show>
```



Handling Null Values

for one feature with null values, I have used SimpleImputer from sklearn in the pipeline to simply impute the mean value. Two reason for the using the simple mean imputer are; 1) There is only one feature in the entire dataset so there is no need to take any complicated measures like using IterativeImputer or Multivariate Imputer, 2) I have used mean value for the imputation because the outliers have already been removed. So no need to use median here.

Lastly, I have standardized the features by removing the mean and scaling to unit variance using StandardScaler to optimize the performance of the ML models.

Results & Conclusion

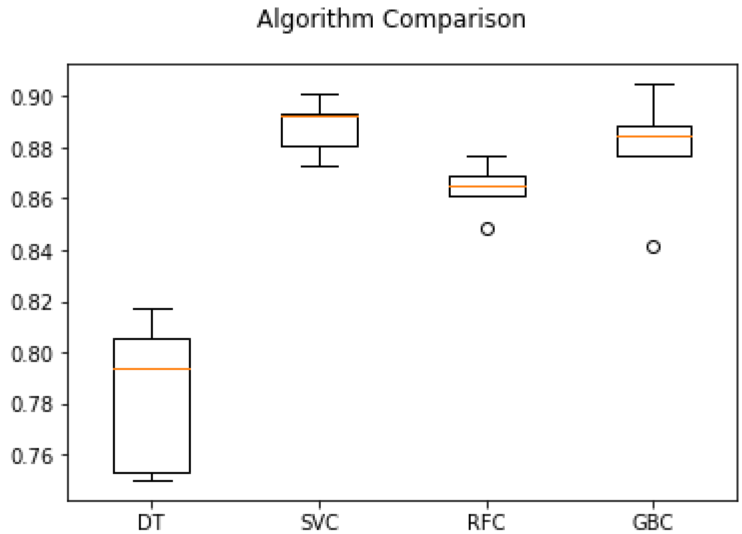
As depicted in the graph below, I have selected Support Vector Classifier with tuned hyperparameters and other transformations (such as Null value imputation and Standardization) in the pipeline for the deployment in the Tosco & Spency store.

We have evaluated four classifiers in our comparative study;

1. Decision Tree (DT)
2. Support Vector Classifier (SVC)
3. Random Forest Classifier (RFC)
4. Gradient Boosting Classifier (GBC)

Among these four, it can be seen that SVC, RFC and GBC are closely competing with each other however, SVC has shown more power and wins my trust for the deployment and implementation on test set.

```
DT: 0.783931 (0.027550)
SVC: 0.888010 (0.009809)
RFC: 0.864166 (0.009385)
GBC: 0.879273 (0.021064)
```



Part 3: Sunsbory's Customer Analysis

Introduction

In this part, I will be predicting not only if the customer is prone to buy few expensive products but also the amount of money that a customer usually spend in a month. Hence it is a regression problem to solve.

Data Preprocessing

We begin by loading the data using Pandas library.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	Target
0	6.57	2	1.42	119.73	-3.34	Low	7.83	-4506.63	-16448.13	-214.64	6.96	-29.12	6	USA	3835.29	660.06	288.54
1	17.64	2	0.70	102.48	-9.77	High	2.43	-3326.25	-15865.93	-199.36	9.20	-30.42	4	UK	4130.94	683.22	1075.23
2	6.06	1	14.72	249.60	-2.26	Very high	4.29	-2206.02	-11705.56	-149.86	12.97	-21.58	16	USA	5305.89	769.83	1722.09
3	2.07	3	0.00	149.85	-0.99	High	3.50	-2798.73	-13815.70	-219.50	5.78	-38.10	10	Europe	2149.47	720.63	3376.78
4	18.99	5	1.92	26.67	-5.62	Low	3.10	-4357.92	-18105.59	-208.86	7.38	-7.06	6	Europe	5115.03	789.96	0.00

While inspecting the data, we see that two features are object and are non-numeric, F6 and F14 respectively. If we notice in the 1st picture above, F6 is a non-numeric ordinal variable hence we can just map ordinal numerics to encode the feature (using LabelEncoder). However, in contrast, the feature F14 is a categorical variable hence I have encoded it using OneHotEncoding technique

Since there were no empty cells or null values found. Hence below picture is the final snapshot of the data after preprocessing.

```
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   F1      1500 non-null   float64
 1   F2      1500 non-null   int64  
 2   F3      1500 non-null   float64
 3   F4      1500 non-null   float64
 4   F5      1500 non-null   float64
 5   F6      1500 non-null   object 
 6   F7      1500 non-null   float64
 7   F8      1500 non-null   float64
 8   F9      1500 non-null   float64
 9   F10     1500 non-null   float64
 10  F11     1500 non-null   float64
 11  F12     1500 non-null   float64
 12  F13     1500 non-null   int64  
 13  F14     1500 non-null   object 
 14  F15     1500 non-null   float64
 15  F16     1500 non-null   float64
 16  Target  1500 non-null   float64
dtypes: float64(13), int64(2), object(2)
```

```
Data columns (total 20 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   F1          1500 non-null   float64
 1   F2          1500 non-null   int64  
 2   F3          1500 non-null   float64
 3   F4          1500 non-null   float64
 4   F5          1500 non-null   float64
 5   F6          1500 non-null   int32  
 6   F7          1500 non-null   float64
 7   F8          1500 non-null   float64
 8   F9          1500 non-null   float64
 9   F10         1500 non-null   float64
 10  F11         1500 non-null   float64
 11  F12         1500 non-null   float64
 12  F13         1500 non-null   int64  
 13  F15         1500 non-null   float64
 14  F16         1500 non-null   float64
 15  Target      1500 non-null   float64
 16  F14_Europe  1500 non-null   uint8  
 17  F14_Rest    1500 non-null   uint8  
 18  F14_UK      1500 non-null   uint8  
 19  F14_USA     1500 non-null   uint8  
dtypes: float64(13), int32(1), int64(2), uint8(4)
```

Identifying and Removing Outliers

For the given data, it did not seem to be a good idea to remove the outliers. As per my analysis using IQR statistical dispersion measure on the data, I got this result:

```
(37, 20)
1463 outliers removed from 1500 with constant 0.5
(441, 20)
1059 outliers removed from 1500 with constant 1.5
(564, 20)
936 outliers removed from 1500 with constant 2.0
```

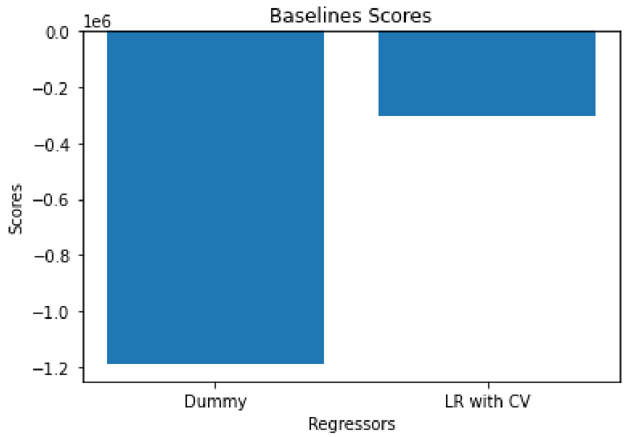
It indicates that the data points in the dataset are quite dispersed with each other. Hence it did not seem to be a good idea to removing the outliers.

Evaluation Metric

I am using <u>Negative Mean Squared Error</u> as the evaluation metric to maximize given the models' hyperparameters. That is, the greater the number is, the smaller the error will be, hence maximizing the metric would result in better performing model.

Baseline Approach

As a baseline approach, I first have simply have applied the some ML models without any hyper parameter tuning and standardizing the data points. I have tried a DummyRegressor and LinearRegressor (with Cross-Validation) to achieve baseline Negative MSE scores.



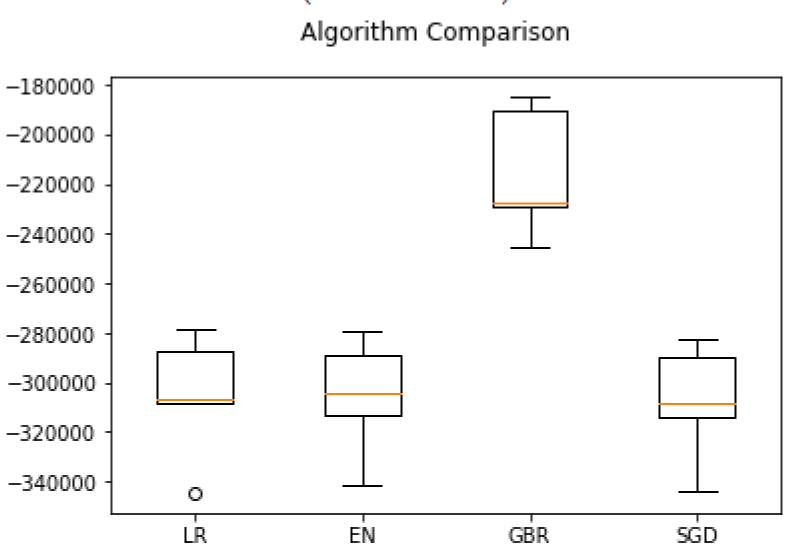
It is good to see that LinearRegressor (with CV) has performed tremendously better than ordinary or common anticipation of DummyRegressor which predicts the mean target value from the fitted data.

Results & Conclusion

We have an <u>outperforming model Gradient Boosting Regressor (GBR)</u> among three other models (that is, Linear Regressor, ElasticNet, and Stochastic Gradient Descent).

GBR has greatest Negative MSE (or smallest error) in comparison to others with a big difference. Hence I choose this pipeline which includes standardization of the datapoints using StandardScaler and then followed by the model for deployment to Sansbury's Customer Analysis requirement.

```
LR: -305033.459494 (22823.338496)
EN: -305418.220829 (21434.017219)
GBR: -215497.891032 (23426.008682)
SGD: -307727.805281 (21379.434941)
```



I have used the best performing pipeline as mentioned above for the test data set. Predictions are attached.