SparkCognition Data Science Assignment

Conventional Data Science approach was followed right from Data exploration to model interpretation.

Decent amount of time was being invested in data exploration and transformation stages.

Data Exploration/ Analysis:

Note: All of the findings along with visualizations are commented in Python notebook as well

Some of the key findings from the data:

Training Dataset (marketing_training.csv) Info**:**

Number of variables/features	22
Number of observations	7414
Missing cells	4670 (2.9%)
Duplicate rows	26 (0.4%)
Total size in memory	1.2 MiB
Average record size in memory	176.0 B

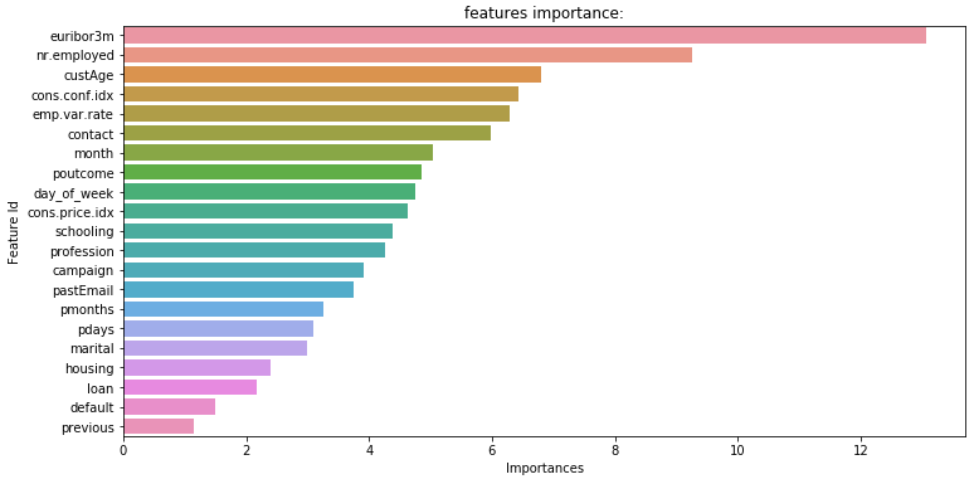
Target Variable:	Responded	**Type:** Boolean		
Unique (%)	< 0.1%	value	count	percentage
Missing (%)	0.0%	no	6574	88.7%
Missing (n)	0	yes	840	11.3%

Dataset has **26 (0.4%)** duplicate rows	Warning(delete these rows)
custAge has **1804 (24.3%)** missing values	Missing (needs treatment)
day_of_week has **711 (9.6%)** missing values	Missing (needs treatment)
euribor3m is highly correlated with emp.var. rate (**ρ = 0.9709545127**)	Rejected(remove one column)
nr. employed is highly correlated with euribor3m (**ρ = 0.9425450982**)	Rejected(remove one column)
pastEmail has **6495 (87.6%)** zeros could be because of binary values	Zeros (needs more analysis)
pmonths is highly correlated with pdays (**ρ = 0.9999934979**)	Rejected (remove one column)
previous has **6350 (85.6%)** zeros could be because of binary values	Zeros (needs more analysis)
schooling has **2155 (29.1%)** missing values	Missing (needs treatment)

Also, it is clear that we have imbalanced dataset and needs to be processed accordingly. I did observe few things while looking at the profiles of who responded ‘yes’:

- They are in high profile jobs
- Attained high education
- Do not have loan

And also ran it through a feature importance test:



I found that euribor3m, nr_employed, custAge, cons.conf.idx contribute highly towards customer's response, perhaps this would help us run data-driven campaigns in order to get response.

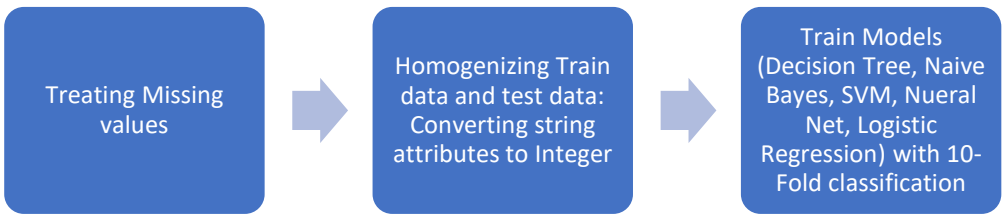
Without further a due, I will dive right into Pre-processing /Modeling

MODEL APPROACH

I have implemented two workflows using classifiers (SVM, Decision Tree, Logistic Regression, Neural net, Random forest)

Workflow 1: Main Idea was to transform categorical variables into Indexed values for example (Categorical values under schooling (undergrad, high school etc.) have all been indexed.

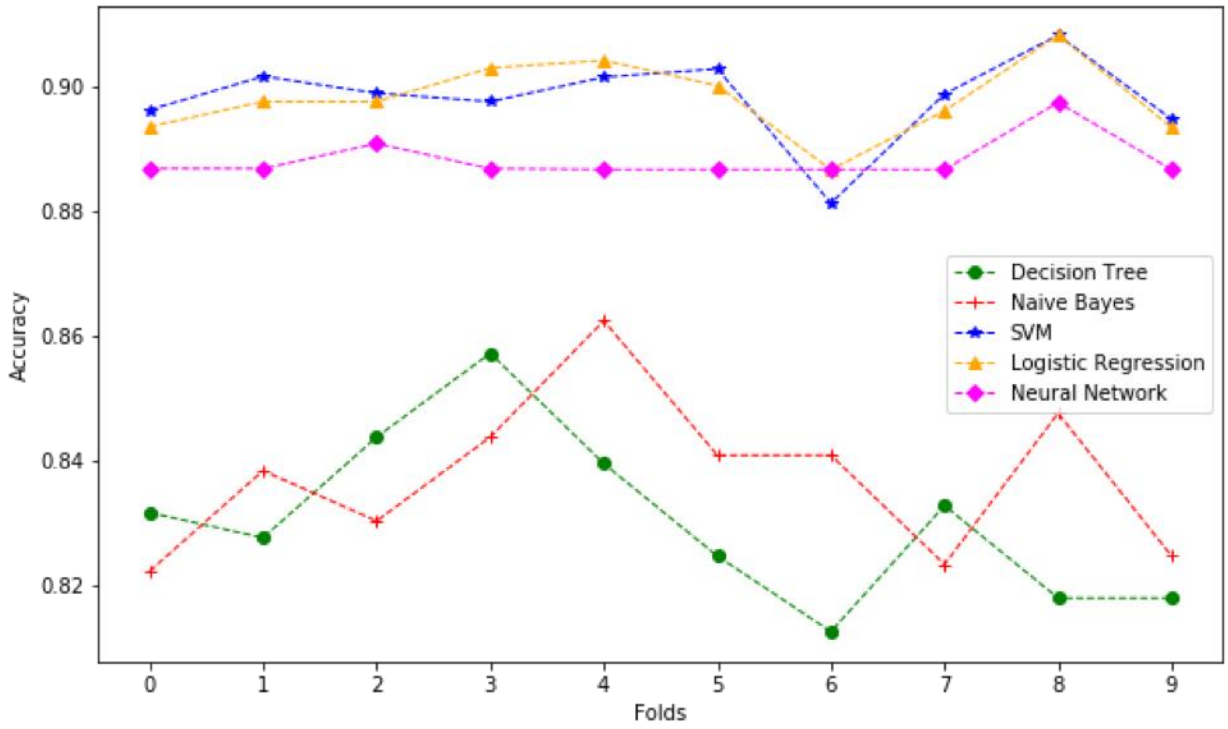
	custAge	profession	marital	schooling	default	housing	loan	contact	month	day_c
0	55	1	0	4	0	1	1	0	2	
1	31	4	1	4	1	1	1	0	5	
2	42	2	1	6	1	1	1	1	1	
3	55	0	2	4	0	2	2	0	5	
4	31	1	2	4	1	2	1	0	1	



1. Describe your model and why did you choose this model over other types of models?
2. Describe any other models you have tried and why do you think this model preforms better?

After training and running cross validations on test data set, I choose to report:

- The results for "SVM" and "logistic regression" classifiers.
- This choice is made through empirical evaluation of the training data using 10-fold cross validation on multiple classifiers. The results of these experiments are summarized in following figure



- I tried Decision Tree, Naive Bayes, SVM, Logistic Regression, and Neural Network.
- It can be observed that among 5 classifiers, SVM and logistic regression show a consistent performance on each fold and also have higher accuracy.
- Note that we do not have labels for testing data, thus the 10-fold cross validation is done using training data only.
- Based on these results, we train a logistic regression and an SVM classifier using the complete training data and then get predictions.
- These predicted values are reported in "result.csv".
- The reason that SVM/Logistic Regression performs better can be traced back to the fact that Naïve Bayes treats all the features as independent ones whereas SVM looks at the interactions between them to a certain degree, so it seems to be treating non-linear data better.
- Intuitively, we know that all the features (variables) are dependent, thus favoring the classifiers that exploit this fact.

3. How did you treat missing data?

- The missing data was replaced by majority value for a particular variable as I am limited by other choices (such as using average) for categorical (textual) variables.
- For numerical values like custAge, euribor3n, I have replaced them with mean values.

4. How did you handle categorical (string) data?

- Categorical data was converted to numerical data by mapping each value for a particular categorical feature to a unique integer.

5. How did you handle unbalanced data?

- In this workflow, I did not handle the unbalanced data with respect to the class label, but I have used encoding and sampling in my next workflow
- There were 6,574 instances for *no* class while only 840 instances for *yes* class.
- Although, we could oversample the later, but it does not make much difference.
- Second option is to utilize class weights during the training process by automatically adjusting weights inversely proportional to class frequencies in the input data.
- However, this can only be done with SVM and Logistic Regression. To maintain fairness while comparing all algorithms, we avoid using any balancing technique for any classifier.

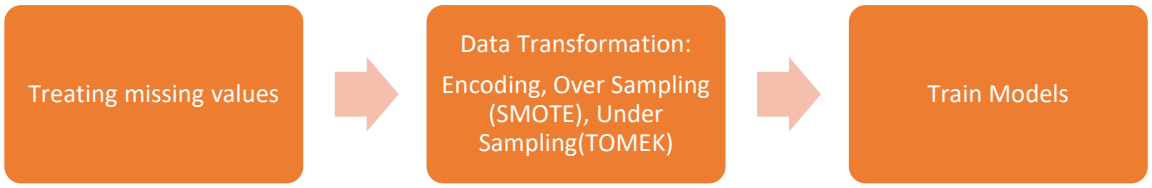
6. How did you test your model?

- I tested the model using 10-fold cross validation on training split as no ground truth is given for testing split that could be used to test the model. However, we submitted the predictions made by each algorithm on testing set as csv file.

This workflow was implemented without much data sampling. **Code and results saved in Workflow 1 directory.**

Next technique we shall see how much data sampling and encoding techniques would reflect our results.

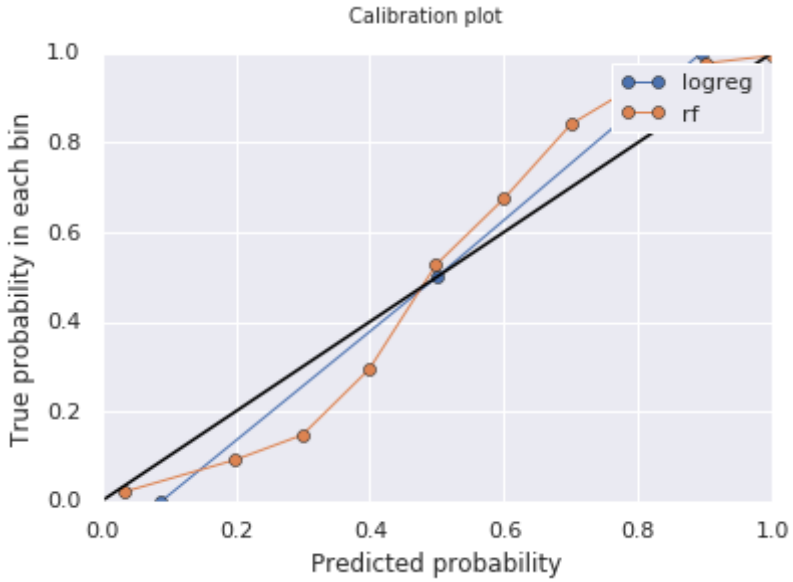
Workflow 2: I have used sampling and encoding techniques before training models, emphasis was more on transformation. Code and results saved in Workflow 2 directory.



1. Describe your model and why did you choose this model over other types of models?
2. Describe any other models you have tried and why do you think this model preforms better?

After training and running cross validations on test data set, I choose to report:

- The results for "SVM", "random forest", "Decision Tree". Both DT and RF are based on rules to decide on the prediction. These two models are simple to explain and gave us the best accuracy among the others.
- This choice is made through empirical evaluation of the training data using 5-fold and 10-fold cross validation on multiple classifiers. The results of these experiments are quantified in following figure:



- We have used logistic regression, Neural Net, SVM. Decision Tree and Random Forest are giving us the best results because are there rules for each splitting of the tree and important features are given preference at time of splitting using entropy.

- Even with encoding and treating imbalance data, we are able to achieve similar scores.
- As we can see Naïve Bayes drastically performs poor

3. How did you treat missing data?

- There are columns with unknown and nonexistent as values. These values can also be treated as NaNs.
- It was evident that customers who were not contacted or failing to contact is filled with values 999 in pdays and pmonths. To solve this, we can create a Boolean variable to show if the customer was previously contacted or not.
- pdays and pmonths are identical in a way they represent the same and must highly correlated. So we need to remove pmonths from the dataset.
- The values in pdays can skew the data and this can be considered outlier as it is too far from the other values. We can substitute 999 with -1 to balance to distribution.
- Imputing missing numerical values with mean.

4. How did you handle categorical (string) data?

- The categorical variables are transformed using one hot encoding. One hot encoding is way to transform the categorical data into binary variables with its unique values becoming columns and filled with 0,1 for non-existent and existent

cons.price.idx	cons.conf.idx	euribor3m	nr.employed	pmonths	...	month_nan	day_of_week_mon	day_of_week_thu	day_of_week_tue	day_of_week_wed
93.200	-42.0	4.191	5195.8	999.0	...	0	1	0	0	0
93.918	-42.7	4.960	5228.1	999.0	...	0	1	0	0	0
93.994	-36.4	4.857	5191.0	999.0	...	0	1	0	0	0
93.918	-42.7	4.962	5228.1	999.0	...	0	0	0	0	1
92.893	-46.2	1.291	5099.1	999.0	...	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...
93.994	-36.4	4.859	5191.0	999.0	...	0	0	0	0	1
93.918	-42.7	4.961	5228.1	999.0	...	0	0	0	1	0
92.893	-46.2	1.250	5099.1	999.0	...	0	0	0	0	0
93.200	-42.0	4.076	5195.8	999.0	...	0	0	1	0	0
93.994	-36.4	4.864	5191.0	999.0	...	0	0	0	0	0

5. How did you handle unbalanced data?

- **Resampling:** A widely adopted technique for dealing with highly unbalanced datasets is called resampling. It consists of removing samples from the majority class (under-sampling) and / or adding more examples from the minority class (over-sampling).

  Techniques involve:

- **Under - sampling: Tomek link**

  Tomek links are pairs of very close instances, but of opposite classes. Removing the instances of the majority class of each pair increases the space between the two classes, facilitating the classification process.

- **Over - sampling: SMOTE**
  SMOTE (Synthetic Minority Oversampling Technique) consists of synthesizing elements for the minority class, based on those that already exist. It works randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.

- **Over-sampling followed by Under-sampling**

6. How did you test your model

I created a function roc_analysis that fits a classifier to a subset of the data and then cross-validates it on the remainder of the data. This fit-and-validate cycle is repeated k times (k-fold cross validation).

The function also produces ROC curves to quantify model performance. In the process I realized, In the presence of skewed classes, ROC curves better characterize algorithm performance than classification accuracy/error.

Conclusion

Aim of the using 2 workflows was to understand and make the model more robust and reproducible. I spent most of the time in Workflow 2 as I was putting much emphasis on reproducible workflow (Created functions for Variable Transformation, Feature standardization, imputing missing values, resampling to deal with imbalanced data, modelling and training). Key insights I found out were:

- Both workflows produced almost the same results given the data sample, so resampling the data did not make much difference. Probably if I were to be provided with more data, we could do better sampling and train the model to make much accurate predictions.

- Few features like euribor3m, nr_employed, custAge, cons.conf.idx contribute highly towards customer's response, so it would be helpful to look at the feature variables when running data-driven campaigns based on customer profiles, probably once we do customer segmentation probably personalized campaigns would benefit the customers as well as the company.