INDEX

(OCTOBER 2020)

CLASSIFICATION OF CLIENTS OF A BANK'S MARKETING CAMPAIGN

PREPARED BY: ARMANDO MEDINA

1. INTRODUCTION

This project is part of the Udacity Azure Machine Learning Nanodegree. In this project, we build and optimize an Azure ML pipeline using the Python SDK and a provided Scikit-learn model. This model is then compared to an Azure AutoML run.

The specific project is based on analyzing the data that we have from the clients to determine if they will subscribe or not to the service offered, which is a term deposit.

2. BUSINESS PROBLEM

Customer acquisition is always a non-trivial problem in any company, regardless of the channel used to acquire customers, capture leads and convert them into customers of the company's products is a task that requires time and money. Therefore, companies would like to be able to predict if a given client will subscribe into a given product offered through a phone call.

Specifically, we explore a set of data related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The problem we want to solve is to predict through the information if a client is going to subscribe or not a term deposit offered through a phone call.

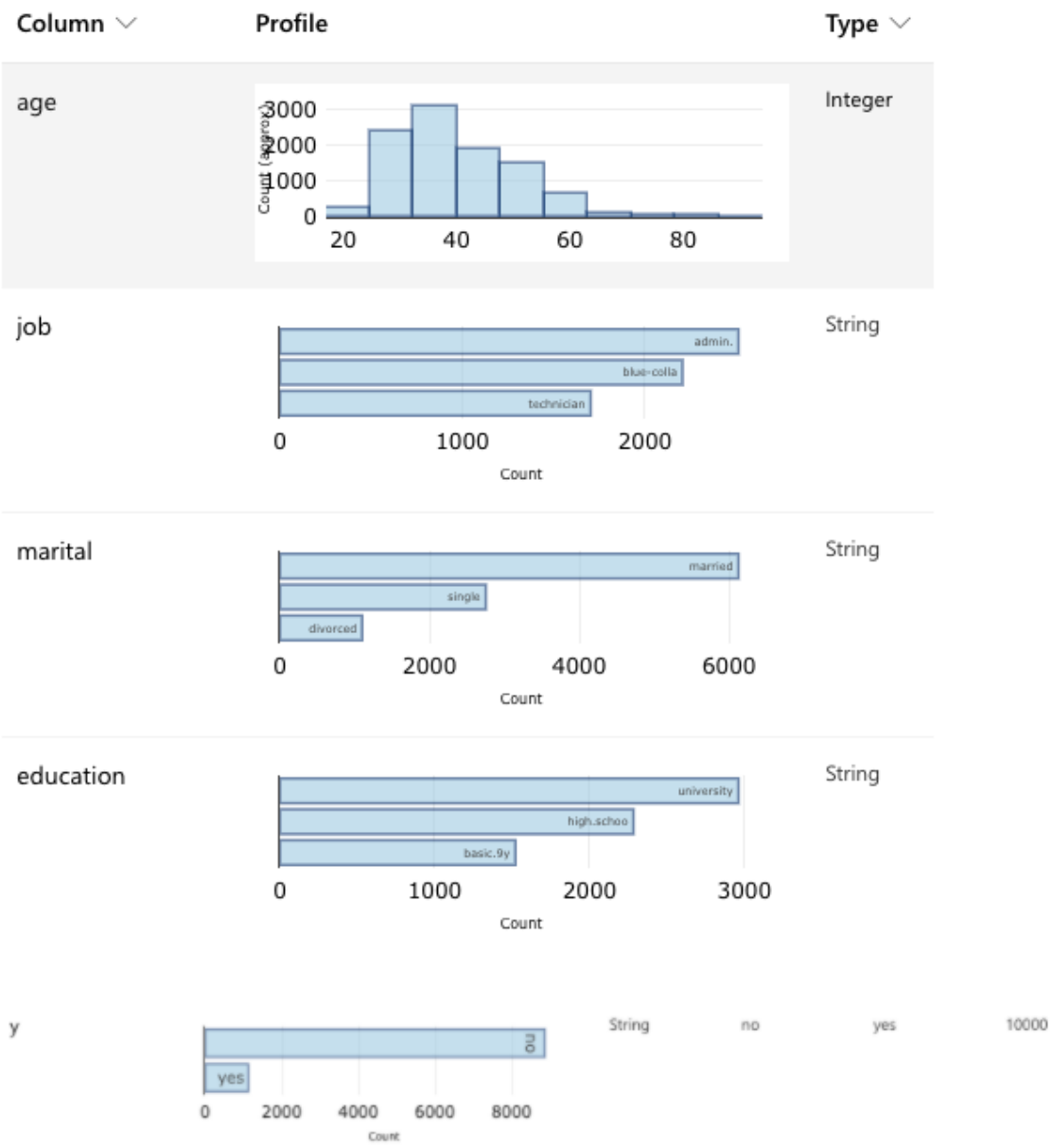
3. DATA

The data is related to direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls.

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe to a term deposit (variable y)

- .For our work we will use the link offered by the Nanodegree that is located at: https://automlsamplenotebookdata.blob.core.windows.net/automl-sample-notebook-data/bankmarketing_train.csv

- A variant of this data set can be found in: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing



4. METHODOLOGY

In this project, our objective is to be able to predict if a client is going to subscribe or not to a long-term deposit. For this, we are going to use a scikit-learn model and with the help of the Python SDK we are going to optimize its hyperparameters through HyperDrive. Then, we are going to apply AutoML to the dataset. inally, we are going to compare the best model thrown from AutoML with the linear regression model optimized with the help of HypeDriver.

In a detailed way, we will follow the following steps:

1. Inside the train.py file, we are going to load our dataset with the help of TabularDatasetFactory.

2. Then, we are going to clean our dataset for better handling, for this we are going to help the clean_data function (located in cleandata.py). With this function, using pandas, we are going to eliminate null values if they exist, and transform some columns as "housing" for a better result of our model.

3. Once the ETL is passed, we are going to divide our data into training data and test data (or validation).

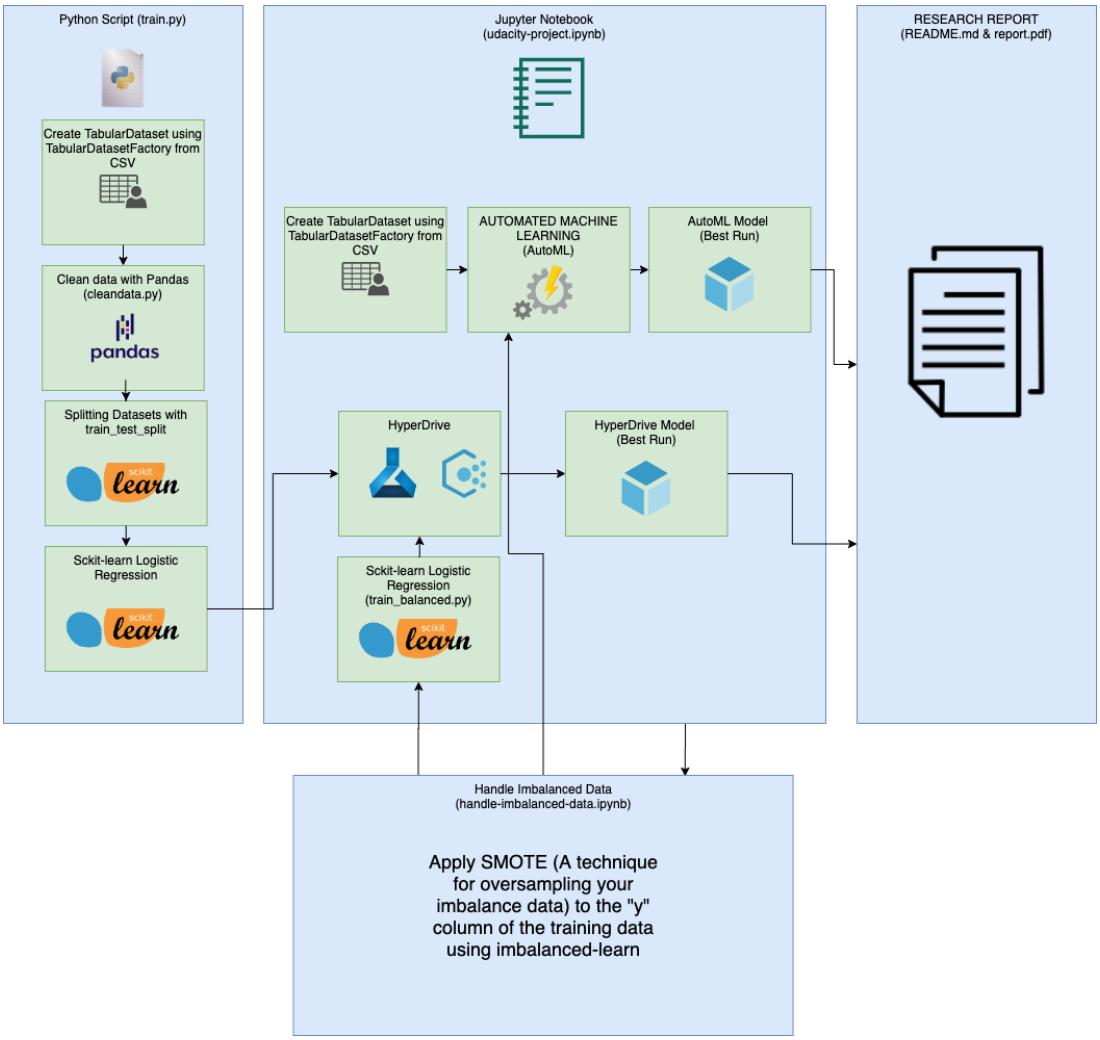
4. In this step, usingAzure ML and HyperDriver, we are going to optimize our model specifically in the parameters of Regularization Strength (C) and Max iterations (max_iter). Also, in our configuration file, we pass our early stopping policy and our estimator with our model.

5. Once HyperDriver finishes optimizing our model, we will register the best model and analyze the results of this compared to the others, for this we will help with accuracy.

6. After the optimization of our Linear Regression model with HyperDrive, we will implement AutoML to our dataset, for this we create an experiment passing it as parameters: our input data, our validation data, the type of task that in this case is the classification (yes or no), the column that we want to predict, which in this case is "y", our metric that in order to compare with the previous model we will use "accuracy", we specify a timeout time and finally, in this case, we will specify two models that we do nott want AutoML to use in its search for the best model for our problem.

7. Once our experiment is finished, we will register the best model and compare it with our previous result.

8. When analyzing the Confusion Matrix of both models we observe that the model has biases due to the imbalance of our dataset. For this, we use the SMOTE technique applied in the "y" column and repeat the experiments already carried out.



5.RESULTS AND DISCUSSION

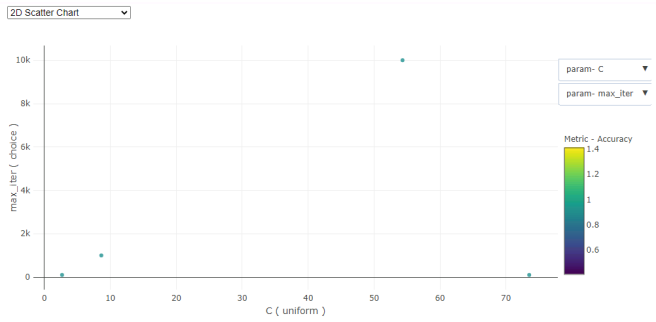
As a result, we can say that there is not much difference between the two final models, although the best model produced by AutoML predicts slightly better.Since the difference is not significant, it must be validated how both models generalize.

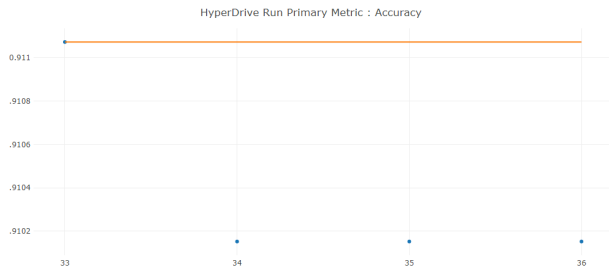
For our model resulting from the optimization of parameters with HyperDrive, we have that the four results offered similar performances with accuracy metrics of 91% and a training execution time between 1:34 - 1:41.

Completed (4)

Run	Best Metric*	Status	Started	Duration	Run Id
33	0.91107228		Oct 26, 2020 9:45 PM	0:01:34	HD_3094fe1b-97b0-438f-9c16-8a512d503875_1
34	0.91015266		Oct 26, 2020 9:45 PM	0:01:38	HD_3094fe1b-97b0-438f-9c16-8a512d503875_3
35	0.91015266		Oct 26, 2020 9:45 PM	0:01:41	HD_3094fe1b-97b0-438f-9c16-8a512d503875_2
36	0.91015266		Oct 26, 2020 9:45 PM	0:01:39	HD_3094fe1b-97b0-438f-9c16-8a512d503875_0

* The best metric field is obtained from the min/max of primary metric achieved by a run





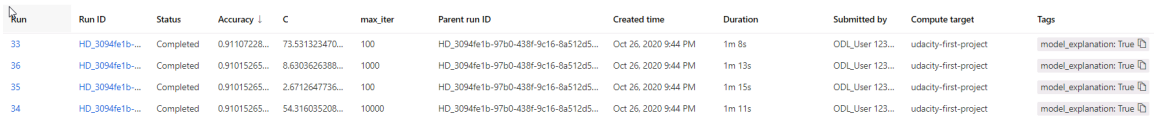
Regarding the early termination policy, it was defined based on slack criteria and a frequency for evaluation. This early termination policy prevents experiments from running for a long time and using resources unnecessarily.

In machine learning, a hyperparameter is a parameter whose value is used to control the learning process.

In the case, the hyperparameters of the best model were the following:

- ***max_iter=100***
  Maximum number of iterations of the optimization algorithm

- ***C= 73.5313***
  Each of the values in C describes the inverse of regularization strength. Like in support vector machines, smaller values specify stronger regularization.



Run	Run ID	Status	Accuracy ↓	C	max_iter	Parent run ID	Created time	Duration	Submitted by	Compute target	Tags
33	HD_3094fe1b-...	Completed	0.91107228...	73.531323470...	100	HD_3094fe1b-97b0-438f-9c16-8a512d5...	Oct 26, 2020 9:44 PM	1m 8s	ODL_User 123...	udacity-first-project	model_explanation: True
36	HD_3094fe1b-...	Completed	0.91015265...	8.6303626388...	1000	HD_3094fe1b-97b0-438f-9c16-8a512d5...	Oct 26, 2020 9:44 PM	1m 13s	ODL_User 123...	udacity-first-project	model_explanation: True
35	HD_3094fe1b-...	Completed	0.91015265...	2.6712647736...	100	HD_3094fe1b-97b0-438f-9c16-8a512d5...	Oct 26, 2020 9:44 PM	1m 15s	ODL_User 123...	udacity-first-project	model_explanation: True
34	HD_3094fe1b-...	Completed	0.91015265...	54.316035208...	10000	HD_3094fe1b-97b0-438f-9c16-8a512d5...	Oct 26, 2020 9:44 PM	1m 11s	ODL_User 123...	udacity-first-project	model_explanation: True

The columns that most influence the prediction of this model:

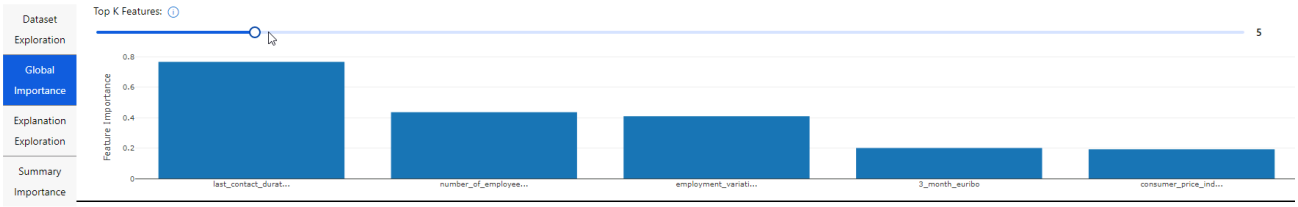
1. Last contact duration: This attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

2. Number of Employees – Quarterly indicator: Number of employed persons for a quarter.

3. Employment variation rate: It refers to cyclical employment variation.

4. Three Month Euribor: Euribor is short for Euro Interbank Offered Rate. The Euribor rates are based on the interest rates at which a panel of European banks borrow funds from one another.

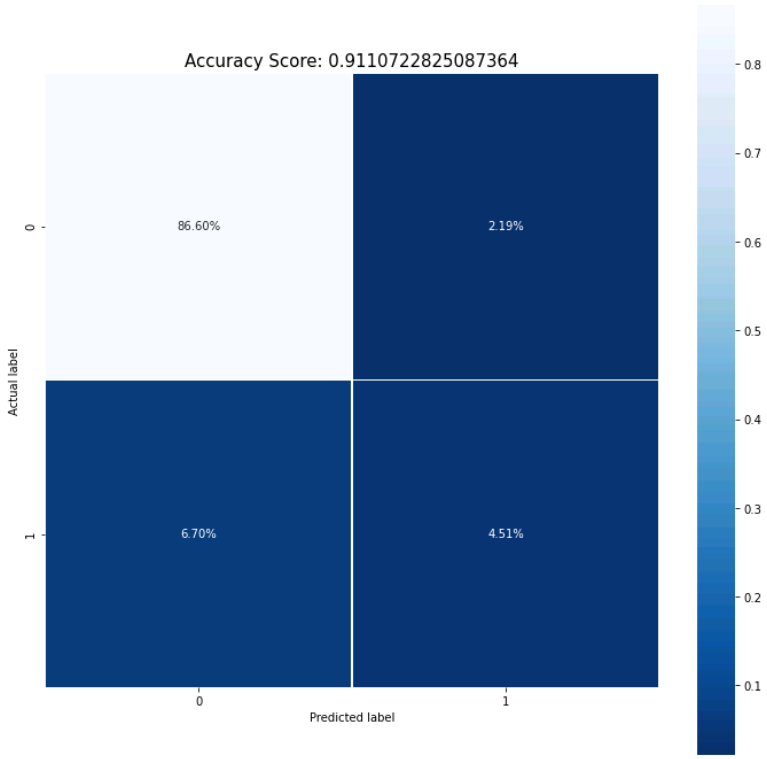
5. Consumer price index: The Consumer Price Index (CPI) is a measure of the average change over time in the prices paid by urban consumers for a market basket of consumer goods and services.



Regarding the performance of our "best_run" we can see that the model manages to classify the "no" very well but still has problems to classify the "yes" correctly.

In detail:

- 98% of the "no" were classified correctly.
- 2% of the "no" were classified as "yes" incorrectly.
- 40% of the "yes" were classified correctly.
- 60% of the "yes" were classified as "no" incorrectly.



Now we are going to analyze our models generated by AutoML.

To begin with, one of the things that called our attention was that AutoML warned us that the dataset had a balance problem which increased the probability of bias, we will see it in detail later.

```
TYPE:         Class balancing detection
STATUS:       ALERTED
DESCRIPTION:  To decrease model bias, please cancel the current run and fix balancing problem.
              Learn more about imbalanced data: https://aka.ms/AutomatedMLImbalancedData
DETAILS:      Imbalanced data can lead to a falsely perceived positive effect of a model's accuracy because the input data has
bias towards one class.
+---------------------------------+---------------------------------+--------------------------------------+
|Size of the smallest class       |Name/Label of the smallest class |Number of samples in the training data|
+=================================+=================================+======================================+
|2944                             |1                                |26425                                 |
```

Automated machine learning, also referred to as automated ML or AutoML, is the process of automating the time consuming, iterative tasks of machine learning model development.

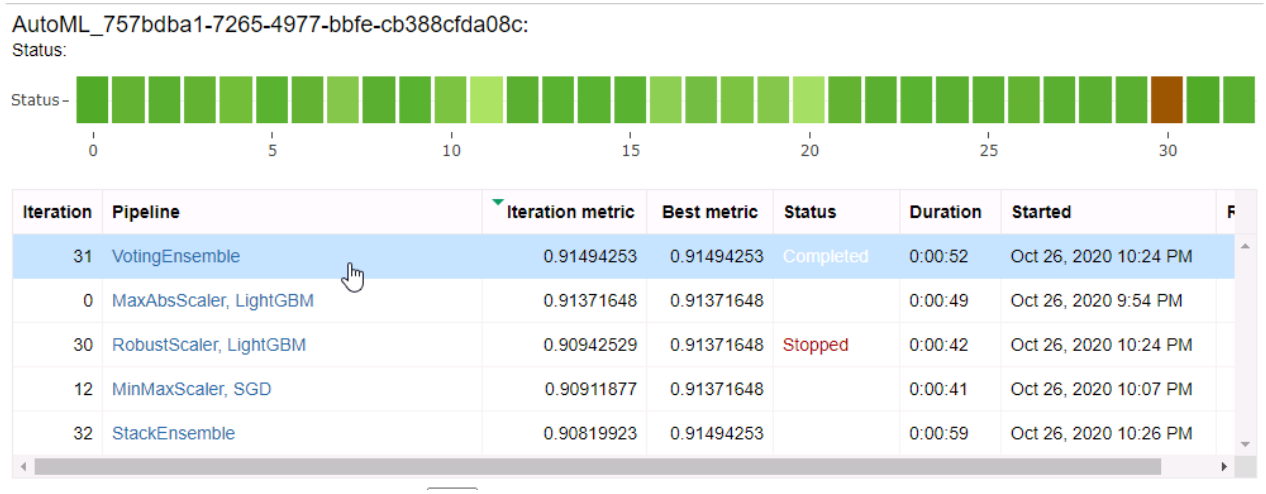
AutoML allows you to train, evaluate, improve, and deploy models based on your data, which allows us to test and discard hundreds of models in the time it would take to test one.

In this particular case AutoML tested the dataset with around 32 different models.

ITERATION	PIPELINE	DURATION	METRIC	BEST
WARNING - Received unrecognized parameter primary_metric_name				
0	MaxAbsScaler LightGBM	0:00:30	0.9137	0.9137
1	MinMaxScaler RandomForest	0:00:30	0.8884	0.9137
2	StandardScalerWrapper SGD	0:00:35	0.9061	0.9137
3	MinMaxScaler RandomForest	0:00:59	0.8854	0.9137
4	StandardScalerWrapper SGD	0:00:31	0.8533	0.9137
5	StandardScalerWrapper RandomForest	0:00:28	0.8984	0.9137
6	RobustScaler ExtremeRandomTrees	0:00:30	0.8912	0.9137
7	StandardScalerWrapper ExtremeRandomTrees	0:00:26	0.8159	0.9137
8	StandardScalerWrapper SGD	0:00:29	0.9067	0.9137
9	StandardScalerWrapper SGD	0:00:28	0.8967	0.9137
10	MinMaxScaler SGD	0:00:33	0.8343	0.9137
11	RobustScaler ExtremeRandomTrees	0:00:31	0.7208	0.9137
12	MinMaxScaler SGD	0:00:29	0.9091	0.9137
13	MinMaxScaler ExtremeRandomTrees	0:00:29	0.8972	0.9137
14	MinMaxScaler ExtremeRandomTrees	0:00:37	0.8973	0.9137
15	MinMaxScaler ExtremeRandomTrees	0:00:29	0.8976	0.9137
16	StandardScalerWrapper RandomForest	0:00:30	0.7910	0.9137
17	StandardScalerWrapper SGD	0:00:29	0.8423	0.9137
18	RobustScaler ExtremeRandomTrees	0:00:29	0.8323	0.9137
19	StandardScalerWrapper RandomForest	0:00:29	0.8116	0.9137
20	MinMaxScaler ExtremeRandomTrees	0:00:27	0.7226	0.9137
21	MaxAbsScaler LightGBM	0:00:33	0.8907	0.9137
22	MinMaxScaler RandomForest	0:00:28	0.9007	0.9137
23	MaxAbsScaler ExtremeRandomTrees	0:00:37	0.8973	0.9137
24	TruncatedSVDWrapper LogisticRegression	0:00:31	0.9051	0.9137
25	SparseNormalizer LightGBM	0:00:31	0.9064	0.9137
26	StandardScalerWrapper ExtremeRandomTrees	0:00:41	0.8854	0.9137
27	MaxAbsScaler SGD	0:00:31	0.9051	0.9137
28	StandardScalerWrapper LogisticRegression	0:00:33	0.9079	0.9137
29	SparseNormalizer RandomForest	0:00:40	0.9011	0.9137
30	RobustScaler LightGBM	0:00:29	0.9094	0.9137
31	VotingEnsemble	0:00:39	0.9149	0.9149
32	StackEnsemble	0:00:45	0.9082	0.9149

For our model resulting from implementing AutoML to our dataset, the precision metrics were between 72% and 91% with an execution time between 0:29 seconds and 0:45 seconds

The best model was the VottingEsemble(which is a set machine learning model that combines the predictions of many other models) followed by the MaxAbsScaler, LightBGM. However both a 91% accuracy similar to our HyperDrive optimized model.



Iteration	Pipeline	Iteration metric	Best metric	Status	Duration	Started
31	VotingEnsemble	0.91494253	0.91494253	Completed	0:00:52	Oct 26, 2020 10:24 PM
0	MaxAbsScaler, LightGBM	0.91371648	0.91371648		0:00:49	Oct 26, 2020 9:54 PM
30	RobustScaler, LightGBM	0.90942529	0.91371648	Stopped	0:00:42	Oct 26, 2020 10:24 PM
12	MinMaxScaler, SGD	0.90911877	0.91371648		0:00:41	Oct 26, 2020 10:07 PM
32	StackEnsemble	0.90819923	0.91494253		0:00:59	Oct 26, 2020 10:26 PM

The hyperparameters used by AuotML in the best model were the following:

- ***max_iter=100***
  Maximum number of iterations of the optimization algorithm

- ***Cs= 10***
  Each of the values in C describes the inverse of regularization strength. Like in support vector machines, smaller values specify stronger regularization.

- ***tol=0.0001***
  Tolerance for stopping criteria.

- ***solver='lbfgs'***
  Algorithm to use in the optimization problem.

- ***penality='l2'***
  Used to specify the norm used in the penalization.

- ***intercept_scaling=1.0***
  Useful only when the solver 'liblinear' is used and self.fit_intercept is set to True. In this case, x becomes, i.e. a "synthetic" feature with constant value equal to intercept_scaling is appended to the instance vector.

The columns that most influence the prediction of this model:

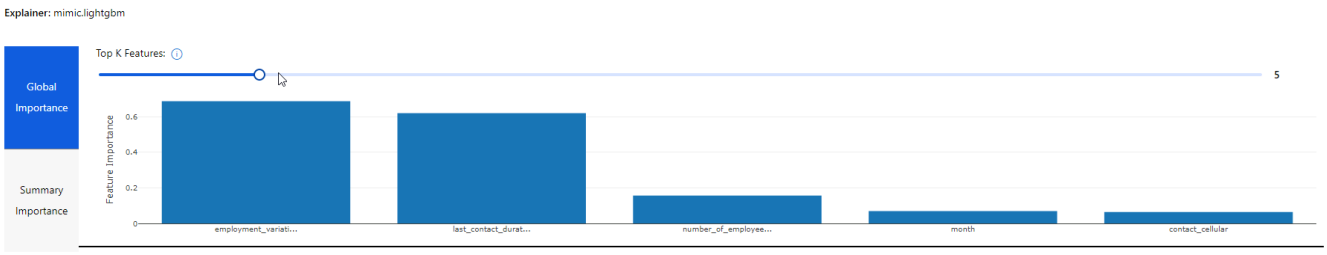
1. Employment variation rate: Is referring to cyclical employment variation.

2. Last contact duration: This attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

3. Number of Employees – Quarterly indicator: Number of employed persons for a quarter.

4. Month: Last contact month of year.

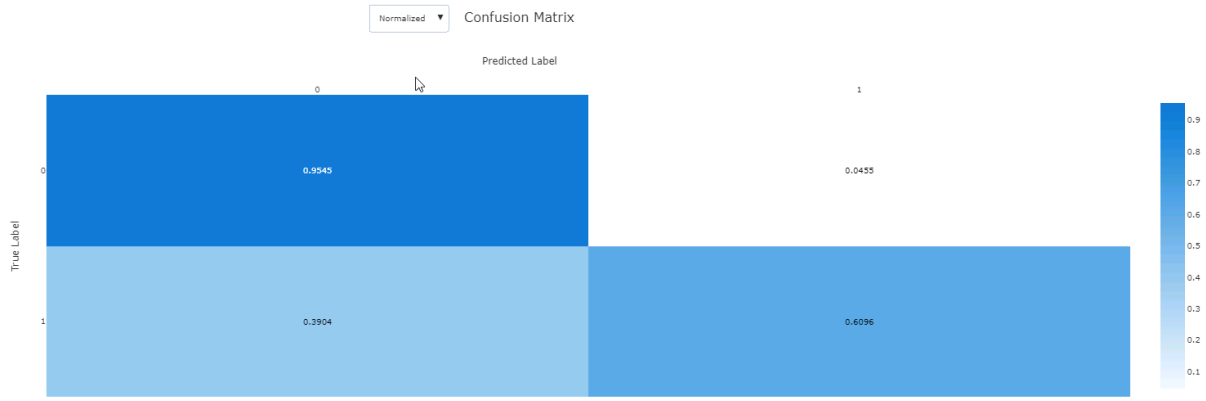
5. Contact Cellular: If the type of communication was through a cell phone.



Regarding the performance of the best model selected by AutoMl, we can see that the classification of the true "no" is improved and the yes is slightly better.

In detail:

- 95% of the "no" were classified correctly.
- 4% of the "no" were classified as "yes" incorrectly.
- 60% of the "yes" were classified correctly.
- 39% of the "yes" were classified as "no" incorrectly.



The benefits of the chosen parameter sampler

Azure Machine Learning supports the following parameter sampling methods:

- **Random sampling:** supports discrete and continuous hyperparameters. It supports early termination of low-performance runs.

- **Grid sampling:** supports discrete hyperparameters. Use grid sampling if you can budget to exhaustively search over the search space. Supports early termination of low-performance runs.

- **Bayesian sampling:** only supports choice, uniform, and quniform distributions over the search space. Bayesian sampling is recommended if you have enough budget to explore the hyperparameter space.

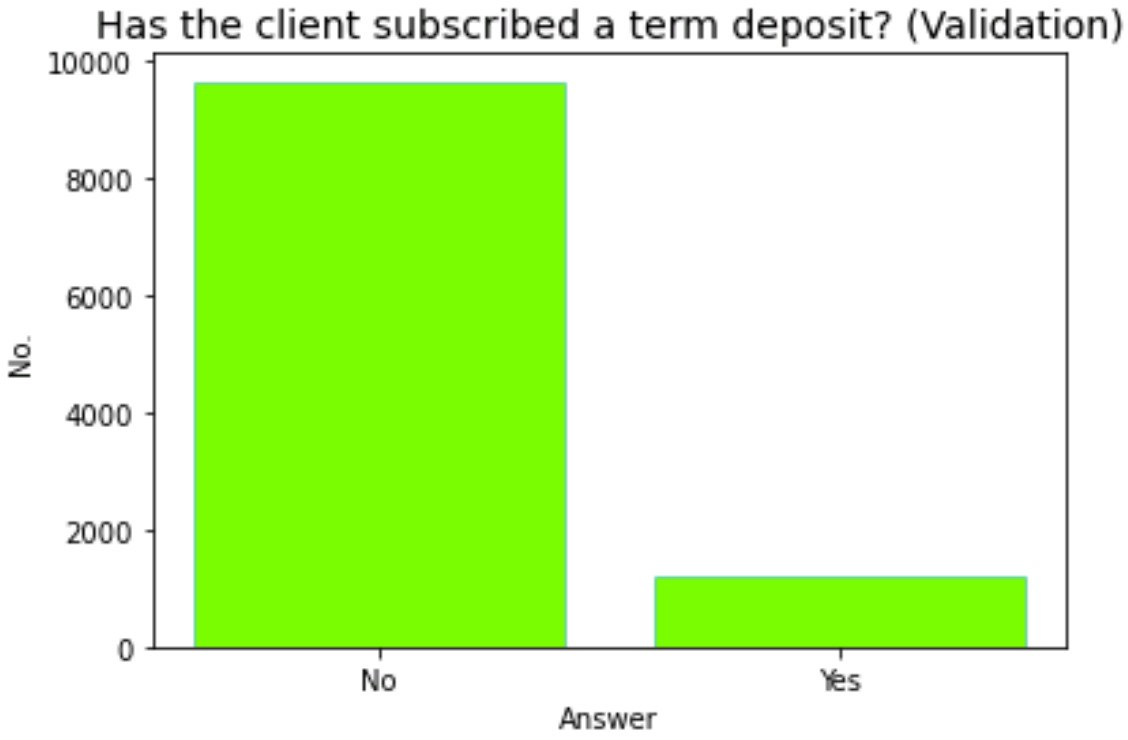
Our selection of ***random sampling*** is motivated because Regularization Strength is a continuous hyperparameter. In other words, random sampling allowed my parameters to be initialized with both discrete and continuous values, and it also allowed for early political termination. This choice gave us an appropriate cost/benefit result.

As a basis for future work, you can read more about the difference between ground sampling and random sampling in the article by James Bergstra and Yoshua Bengio:

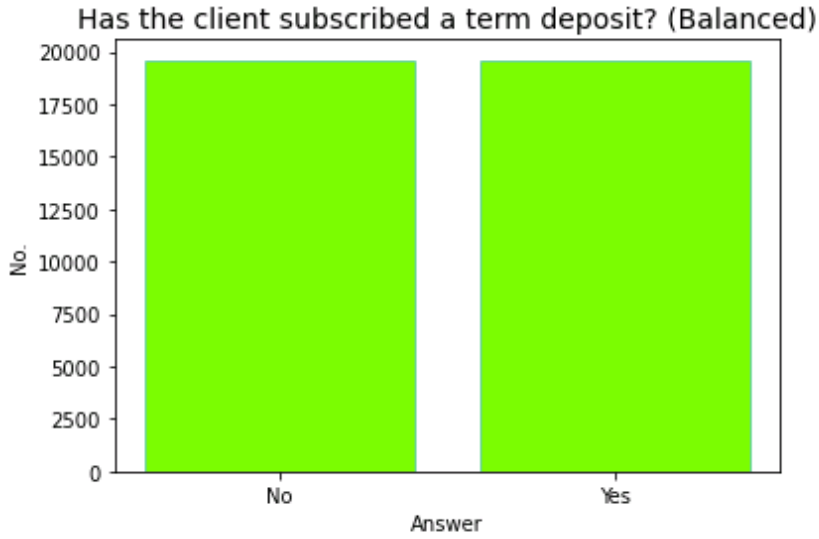
Random Search for Hyper-Parameter Optimization
https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

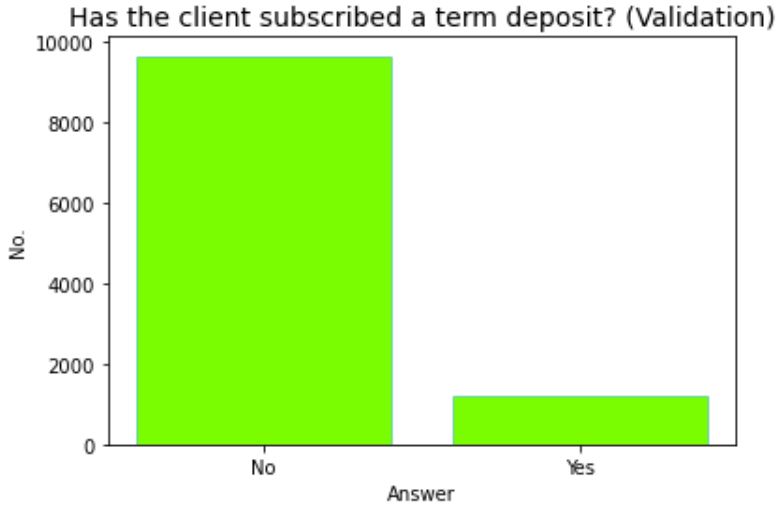
Handle imbalanced data

The variable y is extremely unbalanced, this causes bias, this can be seen in the confusion matrix.



In the handle-imbalanced-data.ipynb notebook included in this project, you can see how the problem is corrected and the dataset is created through the Synthetic Minority Oversampling Technique, or SMOTE for short.

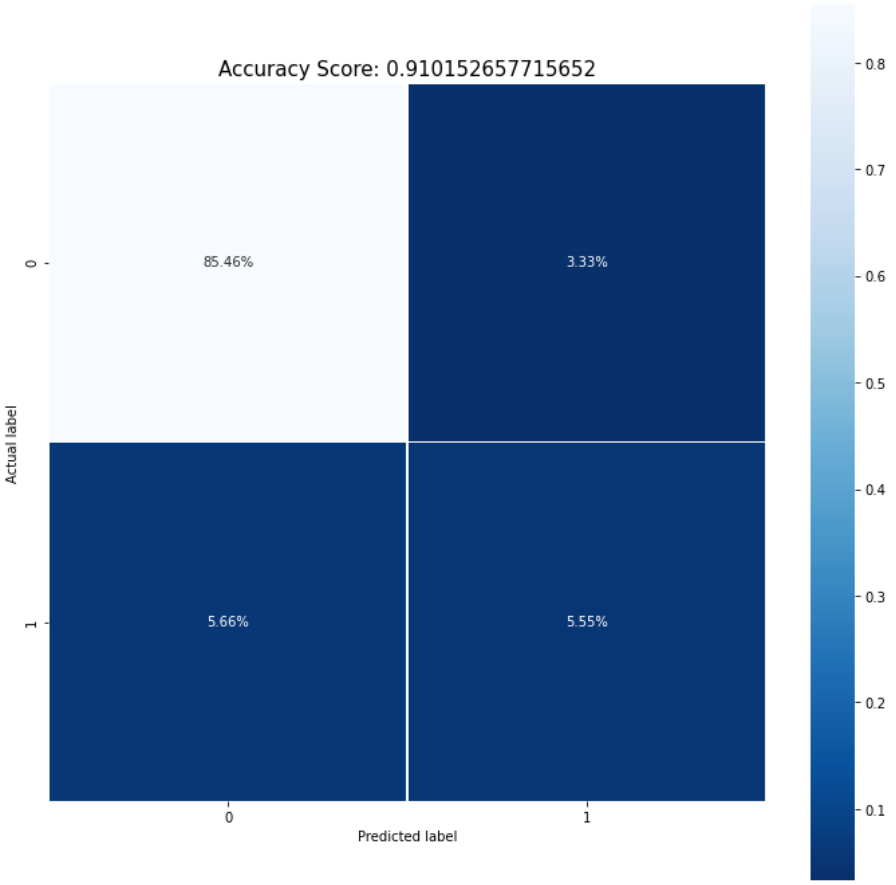




Once the training data was balanced (we left the unbalanced validation date) we ran our experiments for both our experiments.

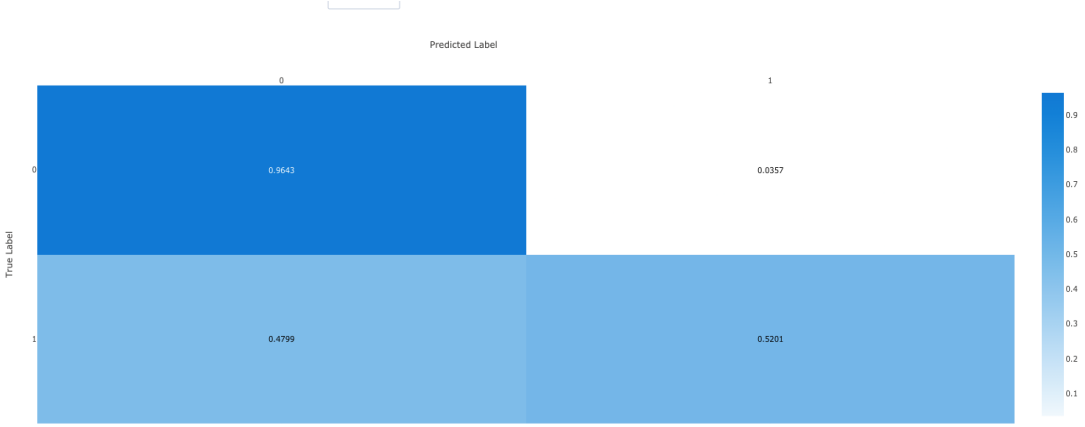
For the best model with hyperparameters optimized with HyperDrive (C:84.379 & MAX_ITER 100):

96% of the "no" were classified correctly.
4% of the "no" were classified as "yes" incorrectly.
49.51% of the "yes" were classified correctly.
50.49% of the "yes" were classified as "no" incorrectly.



For the best AutoML model (l1_ratio=0.3877 & max_iter=1000) :

96% of the "no" were classified correctly.
4% of the "no" were classified as "yes" incorrectly.
52% of the "yes" were classified correctly.
48% of the "yes" were classified as "no" incorrectly.



6.CONCLUSION

The end result of optimizing the hyperparameters with HyperDrive and generating a model with AutoML is quite similar. During the experiments carried out with the dataset, the models gave a prediction of 91% accuracy .

However, this 91% is cheating, mainly because our dataset is imbalanced which produces bias. Realizing this, we applied the SMOTE technique to the column of the customer's response and the results of the models when predicting an if improvement. In the case of the best model, after optimization with HyperDrive, it went from classifying 40% of the "yes" correctly to classifying almost 50% correctly.

However, our dataset is still slightly balanced, especially in two that affect the prediction.

For future work and to obtain better results, two things must be done primarily:

1- Eliminate the ***"last contact duration"*** column in order to bring the models closer to a real-world problem.

2- Correct the balance problem: In every ML project, data management usually represents more than 80% of the work, in this case, there is evidence that more work is needed in the data set, mainly to correct the imbalance. Unbalanced data can lead to a falsely perceived positive effect of a model's precision because the input data is biased towards one class.

3- For future work, it would be interesting to apply Hypedriver to the five best AutoML result models, in addition, it would also be interesting to test Hypedriver with other parameters such as "tol", "solver" and "penalty" that AutoML used during the selection of your model.

7.PROOF OF CLUSTER CLEAN UP