

# Bank Marketing Data Science Exploration

Advanced Data Science Course - Capstone Project

Sasha Lazarevic, Dec 2019

<https://www.linkedin.com/in/LZRVC/>

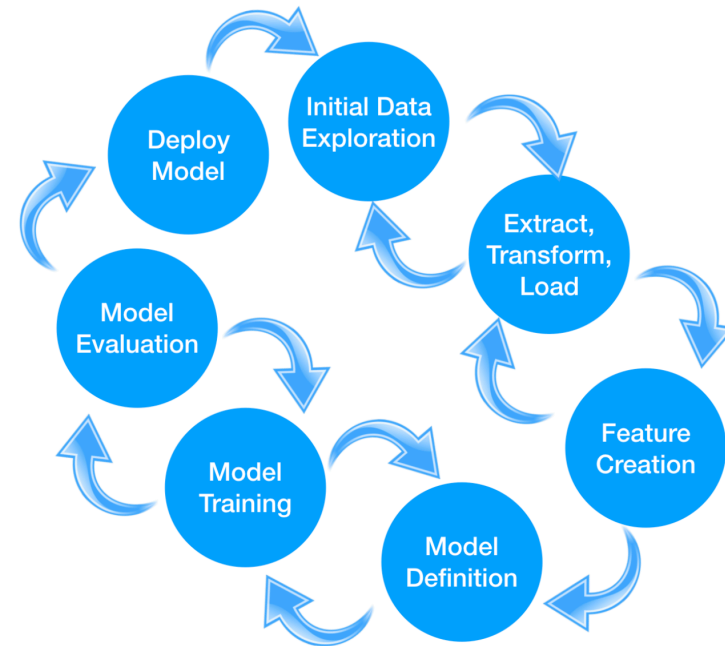
Data Scientist's Deck

# Problem Statement

- A Bank would like to understand which customers will accept the bank's offer for a loan. A realistic dataset has been gathered as presented with 21 columns and 41188 rows, which can be used to train our model.
- Some customers have accepted the offer, but majority not. What features are important ? Can we generate or imagine to add some new features that will improve the quality of our prediction? What algorithm would be the best?

# Methodology

- My data science project will follow a standard data science project methodology with phases like
  - Initial data exploration
  - Extract, Transform, Load
  - Feature Creation
  - Model definition
  - Model training
  - Model evaluation
  - Model deployment




The Lightweight IBM Cloud Garage Method for Data Science Process Model. Source: IBM Corporation

# Toolset

- I will use Watson Studio as Data Science platform with the following tools:
  - Data Refinery, with which I will explore my data set and try to combine data and add some features
  - Jupyter Notebook , which which I will explore the data, and develop machine learning models
  - Watson Machine Learning, which will allow me to train and deploy the models
  - AutoAI, which will allow me to automatically train the models and optimize the hyperparameters

# Dataset

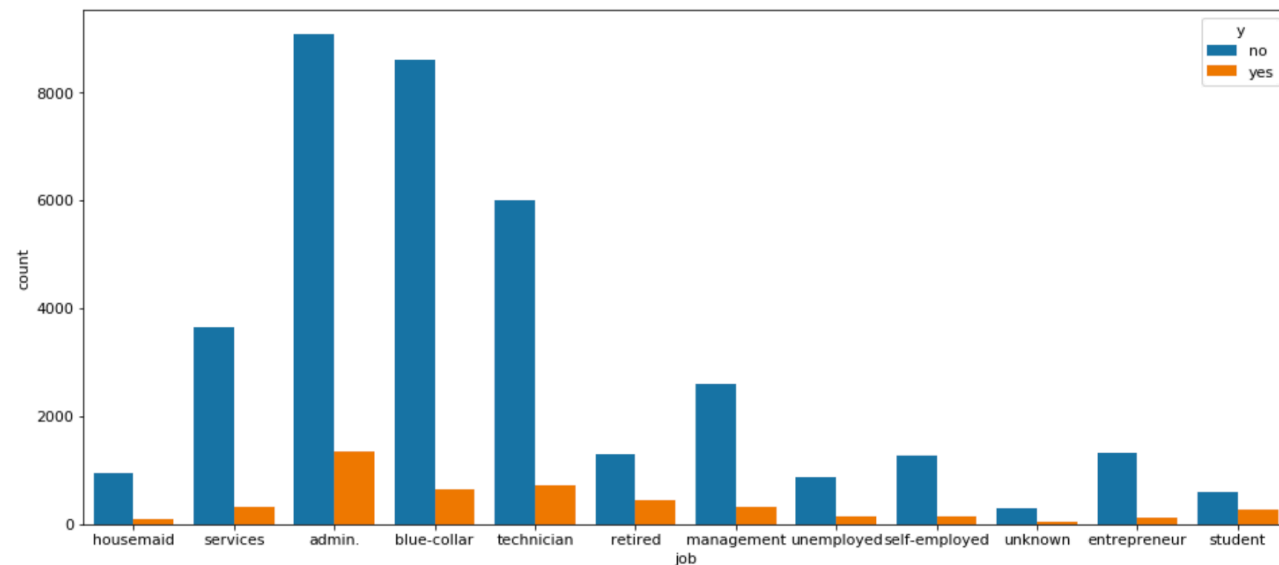
- Dataset is in csv format, uploaded in my Watson Studio Project. Here is the description of the columns
- 1 - age (numeric)  
2 - job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')  
3 - marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)  
4 - education (categorical: basic.4y, 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')  
5 - default: has credit in default? (categorical: 'no', 'yes', 'unknown')  
6 - housing: has housing loan? (categorical: 'no', 'yes', 'unknown')  
7 - loan: has personal loan? (categorical: 'no', 'yes', 'unknown')
- related with the last contact of the current campaign:  
8 - contact: contact communication type (categorical: 'cellular', 'telephone')  
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')  
10 - day\_of\_week: last contact day of the week (categorical: 'mon', 'tue', 'wed', 'thu', 'fri')  
11 - duration: last contact duration, in seconds (numeric). Important note: 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.
- other attributes:  
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)  
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)  
14 - previous: number of contacts performed before this campaign and for this client (numeric)  
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')
- social and economic context attributes  
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)  
17 - cons.price.idx: consumer price index - monthly indicator (numeric)  
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)  
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)  
20 - nr.employed: number of employees - quarterly indicator (numeric)
- Output variable (desired target):  
21 - y - has the client subscribed a term deposit? (binary: 'yes', 'no')

Data assets		
0 asset selected.		
<input type="checkbox"/>	NAME	
<input type="checkbox"/>		BankMarketingCombined
<input type="checkbox"/>	CSV	bank-full.csv
<input type="checkbox"/>	CSV	bank-data-full-mod.csv
<input type="checkbox"/>	XLS	bank-revenue-estimates.xlsx
<input type="checkbox"/>	CSV	bank-additional-full-mod.csv

*bank-additional-full-mod.csv* is the main dataset  
*bank-data-full-mod.csv* is the dataset augmented artificially with an experimental estimate about the clients' revenues

# Data Exploration

- Exploration of the data , shape, number and types of columns
- Exploration of output labels, categories and counts per category
- Exploration of some features and if we can gather any intuition about their influence on the prediction outcome
- Visualization of different features in the context of outcomes



One of the graphics describing the features

# Data Refinery

- Data Refinery is a tool inside Watson Studio which helps Data Scientists visualize and refine the data
- It includes a menu with a set of operations or additional operations can be specified programatically
- Operations are then stored in Refinery Flows, with individual steps that can be scheduled to execute as automatic jobs
- For Bank Marketing Data Science Exploration project, I have created the following Refinery Flow with 6 steps, and using this flow have generated a new dataset that I imported into my Jupyter Notebook and AutoAI

## DATA REFINERY FLOW DETAILS

### LOCATION

Bank Marketing Data Science Explorations

### DATA REFINERY FLOW NAME

**bank-additional-full-mod.csv\_...**

*Enter a description of the Data Refinery flow*

### STEPS

6

## DATA REFINERY FLOW OUTPUT

### LOCATION

Bank Data Science/Data assets

### DATA SET NAME

**bank-data-full-mod.csv**

[More](#) ▾

File format: **CSV**

## Steps

6 STEPS

### Data Source

bank-additional-full-mod.csv

### Convert column type

Automatically converted one or more columns to inferred data types.

### Custom code

```
mutate(age_group=case_when(age <28 ~  
  "22-27", age < 36 ~ "27-35", age<45 ~  
  "36-44", age<59 ~ "45-58", TRUE ~  
  "older than 58"))
```

### Custom code

```
select(age, age_group, everything())
```

### Join

left-joined data from bank-revenue-estimates.xlsx based on columns job,job

### Custom code

```
mutate(revenue_est =  
  case_when(job=="entrepreneur" ~  
    60000, job=="retired" ~ 20000,  
    job=="unemployed" ~ 20000, TRUE ~  
    round(revenues)))
```

### Custom code

```
select(age:education, revenue_est,  
  everything(), -revenues)
```

# Feature Generation

- First, I have created an additional column: AGE\_BINNED

```
In [16]: BMDSE_bafmv1['age_binned'] = pd.cut(BMDSE_bafmv1['age'], 8)
BMDSE_bafmv1['age_binned'].value_counts()
```

```
Out[16]: (27.125, 37.25]      16553
(37.25, 47.375]      11547
(47.375, 57.5]       7641
(16.919, 27.125]      3215
(57.5, 67.625]       1696
(67.625, 77.75]       345
(77.75, 87.875]       157
(87.875, 98.0]        34
Name: age_binned, dtype: int64
```

- I have also added *revenues\_est*

Using Data Refinery, I have combined the data from 'bank-additional-full-mod.csv' and 'revenue\_estimates-csv' to add additional information about estimated revenues of different customers. I expect this to help with prediction

```
In [37]: body = client_e7767fb48b854af082594cf9b01109c0.get_object(Bucket='bankdatascience-donotdelete-pr-amdicwco0zoreh',Key='
data_asset/bank-data-full-mod_7c41c9b3.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

BMDSE_bafmv2 = pd.read_csv(body)
BMDSE_bafmv2.head()
```

Out[37]:

	age	age_group	job	marital	education	revenue_est	default	housing	loan	contact	...	campaign	pdays	previous	poutcon
0	50	45-58	management	married	university.degree	90000.0	unknown	no	yes	telephone	...	1	999	0	nonexiste
1	44	36-44	management	divorced	university.degree	90000.0	no	yes	no	telephone	...	1	999	0	nonexiste
2	41	36-44	management	married	basic.6y	90000.0	no	no	no	telephone	...	2	999	0	nonexiste
3	52	45-58	management	married	university.degree	90000.0	no	no	no	telephone	...	1	999	0	nonexiste
4	57	45-58	management	married	university.degree	90000.0	no	no	no	telephone	...	1	999	0	nonexiste



# Feature Generation

- But these two new features were not so efficient as the artificially generated features that I got with AutoAI

New Feature	Original Feature	Transformation
NewFeature_5	euribor3m	$\tan(\text{euribor3m})$
NewFeature_8	euribor3m	$\tan(\sqrt{\text{euribor3m}})$
NewFeature_0	duration	$\sqrt{\text{duration}}$
NewFeature_3	euribor3m	$\sqrt{\text{euribor3m}}$
NewFeature_9	age	$\text{square}(\text{age})$
NewFeature_4	cons_conf_idx	$\tan(\text{cons\_conf\_idx})$

New Feature	Original Feature	Transformation
NewFeature_10	duration	$\text{square}(\text{duration})$
NewFeature_1	campaign	$\sqrt{\text{campaign}}$
NewFeature_2	pdays	$\sqrt{\text{pdays}}$
NewFeature_7	cons_price_idx	$\tan(\sqrt{\text{cons\_price\_idx}})$
NewFeature_18	cons_price_idx	$\text{square}(\tan(\text{cons\_price\_idx}))$
NewFeature_13	euribor3m	$\text{square}(\text{euribor3m})$

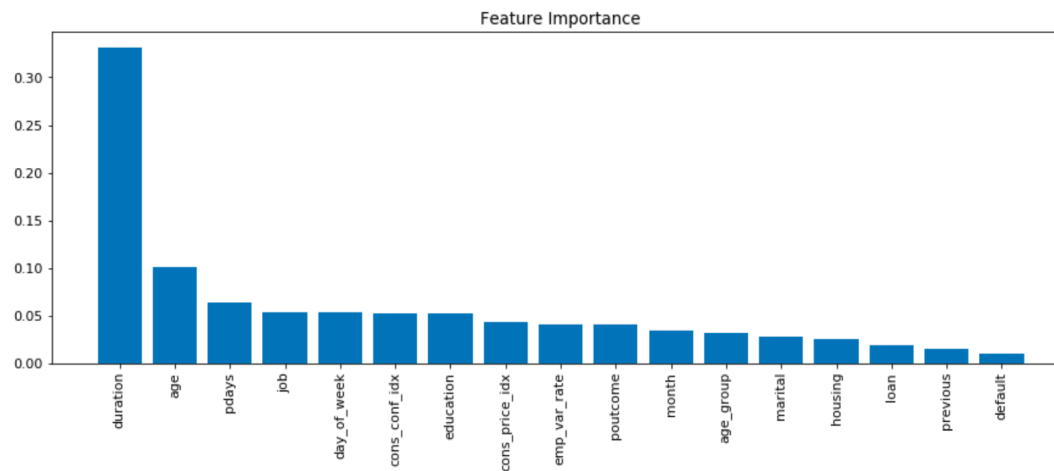
New Feature	Original Feature	Transformation
NewFeature_12	cons_price_idx	$\text{square}(\text{cons\_price\_idx})$
NewFeature_11	pdays	$\text{square}(\text{pdays})$
NewFeature_17	euribor3m	$\text{square}(\sqrt{\text{euribor3m}})$
NewFeature_14	duration	$\text{square}(\sqrt{\text{duration}})$
NewFeature_15	campaign	$\text{square}(\sqrt{\text{campaign}})$
NewFeature_19	cons_price_idx	$\text{square}(\tan(\sqrt{\text{cons\_price\_idx}}))$

# Model Selection

- As the data is structured, I have used ensemble of learners and compared them with deep learning:
  - Random Forest
  - XGBoost
  - LightGBM
  - Keras-based deep neural network (Sequential model)

# Model Training

- I have converted categorical data into numeric and scaled the data from 0 to 1.
- I have splitted the dataset into training (70% ) and test (30% )
- RandomForest has a method to display the importances which is quite useful to see what features contribute the most to the predictions
- Later I have trained XGBoost and LightGBM, and Keras sequential model with two layers and 344 parameters



# Model Training

```
In [74]: model = Sequential()
model.add(Dense(8, input_dim=17, activation='tanh'))
model.add(Dense(16, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
print(model.summary())

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=[auc_roc])

model.fit(np.array(X_train), np.array(y_train), epochs=5, batch_size=128, shuffle=True)

scores = model.evaluate(np.array(X_test), np.array(y_test))
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

Layer (type)	Output Shape	Param #
dense_43 (Dense)	(None, 8)	144
dense_44 (Dense)	(None, 16)	144
dense_45 (Dense)	(None, 1)	17

Total params: 305  
Trainable params: 305  
Non-trainable params: 0

```
None
Epoch 1/5
28831/28831 [=====] - 112s 4ms/step - loss: 0.4510 - auc_roc: 0.4264
Epoch 2/5
28831/28831 [=====] - 118s 4ms/step - loss: 0.2701 - auc_roc: 0.6722
Epoch 3/5
28831/28831 [=====] - 107s 4ms/step - loss: 0.2580 - auc_roc: 0.7437
Epoch 4/5
28831/28831 [=====] - 110s 4ms/step - loss: 0.2555 - auc_roc: 0.7723
Epoch 5/5
28831/28831 [=====] - 105s 4ms/step - loss: 0.2539 - auc_roc: 0.7885
12357/12357 [=====] - 99s 8ms/step

auc_roc: 79.66%
```

# Metrics

- I've used ROC and Area Under Curve for the performance metrics

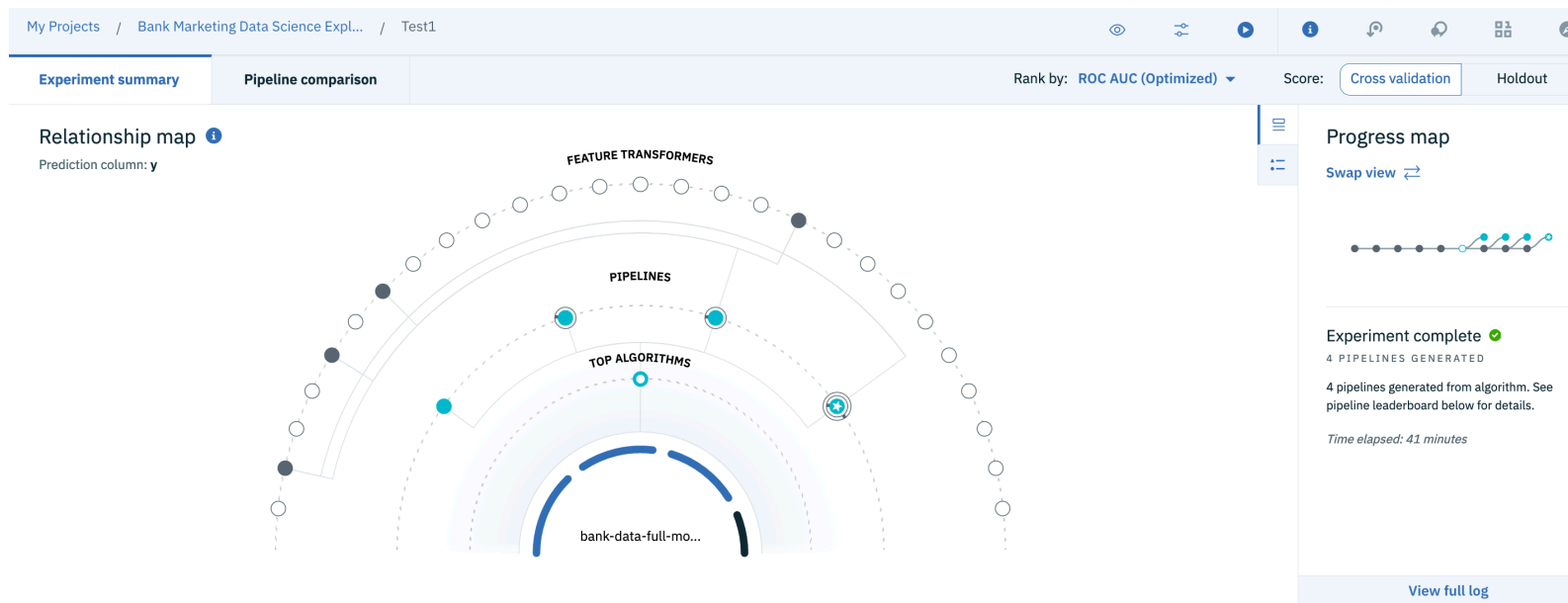
```
In [9]: BMDSE_bafmv1.y.value_counts()
```

```
Out[9]: no      36548  
       yes      4640  
       Name: y, dtype: int64
```

We can see that y has more 'no' than 'yes' answers. This will influence the choice of our performance metrics - ROC is better metric than accuracy in cases where the number of records per class is not balanced

# AutoAI





- AutoAI is the latest functionality of Watson Studio , which allows automatic machine learning of a set of models, automatic selection of the best performing model, automatic hyperparameter tuning and feature generation



# Models Comparison

- The best performing algorithm is LightGBM with ROC AUC better than what I could obtain with my selection of RandomForest, XGBoost or Keras Sequential model

Pipeline leaderboard

	Rank	↑	Name	Algorithm	ROC AUC (Optimized)	Enhancements	Build time
>	★ 1		<a href="#">Pipeline 4</a>	 LGBM classifier	0.951	HPO-1 FE HPO-2	00:25:28
>	2		<a href="#">Pipeline 3</a>	 LGBM classifier	0.951	HPO-1 FE	00:08:55
>	3		<a href="#">Pipeline 2</a>	 LGBM classifier	0.950	HPO-1	00:04:38
>	4		<a href="#">Pipeline 1</a>	 LGBM classifier	0.949	None	00:00:08

## Scores:

- RandomForest ROC AUC score: 86.28%
- XGBoost ROC AUC score: 87.11%
- LGBost ROC AUC score: 91.57%
- Keras Sequential ROC AUC score: 79.66%
- AutoAI LightGBM best ROC AUC score: 95.1%

# Model Deployment

- The best performing model coming from AutoAI is saved as *BMDSE\_LGBMClassifierEstimator\_v4* :

Models				
Watson Machine Learning models				Import model +
NAME	TYPE	RUNTIME	LAST MODIFIED ▾	ACTIONS
BMDSE_LGBMClassifierEstimator_v4	wml-hybrid_0.1	hybrid_0.1	13 Dec 2019	

The model has been successfully deployed in Watson Machine Learning cloud service, which allows to use APIs in the application for the prediction in production

IBM Watson Studio

Upgrade

2013510 - Sasha Lazarevic's...

SL

My Projects

/ Bank Marketing Data Science Expl...

/ BMDSE\_LGBMClassifierEstimator...

/ BMDSE-Deployment01

BMDSE-Deployment01

Overview

Implementation

Test

Implementation

View API Specification

Scoring End-point	<a href="https://us-south.ml.cloud.ibm.com/v4/deployments/c30e39c1-95ea-442d-a444-fa088c87e695/predictions">https://us-south.ml.cloud.ibm.com/v4/deployments/c30e39c1-95ea-442d-a444-fa088c87e695/predictions</a>
Authorization: Bearer <token>	Review the <a href="#">WML authentication</a> documentation for details about generating IAM tokens.
ML-Instance-ID	The "ML-Instance-ID" HTTP header must be populated with the WML instance id, which can be obtained as <a href="#">described here</a> .
Content-type: application/json	Required if the request body is sent in JSON format.

Code Snippets

cURL

Java

JavaScript

Python

Scala

```
# TODO: manually define and pass values to be scored below
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: Bearer $IAM_TOKEN' --header 'ML-Instance-ID: $ML_INSTANCE_ID'
```



# Conclusions

- Using standard data science methodology is key to success of data science projects
- Dataset needs to be of good quality and realistic
- Data Scientist needs to spend time analyzing and exploring the dataset in order to understand the features that could be used for the classification
- Chose of the data science platform and tools is very important for the success of the project and for the productivity of the data scientist
- Deep Learning is not suitable for all types of AI or data science problems. Structured data in tabular format can benefit from traditional machine learning algorithms like Random Forest , XGBoost or LightGBM
- LightGBM proved to be more performant than Random Forest or XGBoost
- AutoAI functionality of Watson Studio is very performant and can save a lot of time to Data Scientist
- Watson Studio with Watson Machine Learning can also be used for deployments, which helps integrate and accelerate the whole process of data exploration, model development and deployments



СПАСИБО **THANK YOU**  
**ХВАЛА !** Grazie  
MERCİ **DANKE** 谢谢

Sasha Lazarevic, IBM Switzerland

<https://www.linkedin.com/in/LZRVC/>  
LZRVC.com