LEVERAGING ML ALGORITHMS FOR **ACCELERATED GROWTH**

CROSS-SELLING RIGHT USING AN ML BASED RECOMMENDATION ENGINE

(USING XGBOOST ALGORITHM)

If you are a banker, you must have thought about ways to push more products to more customers at least once today. Can we do more campaigns? Can we push the relationship managers a little more? Can the on-ground branch staff engage walk-ins? Should we focus on Insurance, a high margin product but difficult to sell or an easy kill like a term deposit? How do we do all of this without irritating the customer?

Cross-selling and Upselling are words every banker hears on a daily basis. And yet, for generations, bankers targeted customers based on walk-ins. Then came excel, and marketing managers started baiting the most profitable customers for insurance or customers with highest savings account balance for a term deposit. It worked, but at the expense of both time and wasted campaign cost.

Recommendation systems are systems that are basically used for recommending next best products/offers to the users either by the user-similarity approach or by using historical data to predict the product/services having highest likelihood to purchase by the user.

What if you could find prospects that not only will buy the product but even appreciate your attempt of selling it to them? What would that mean? Happier customers and happier bankers! That's where Machine Learning (ML) recommendation systems come in.

These systems are widely used in E-commerce platforms, Social media, OTT websites etc. to identify and recommend the correct content/product to its users. The recommendation system usually deals with huge amounts of data to identify the most important factors based on the data provided by the user. The algorithm then identifies the relationship between the user and the items to predict the best product/offer. The customer database has the potential to produce detailed insights and can help in targeting a set of customers for a particular product or recommending a product to a right segment of customer.

In this whitepaper, a recommendation system (classification based) has been created over an open- source banking dataset which contains customer data. The objective is to identify if the user will purchase a term deposit or not. The machine learning algorithm used in the study is Extreme Gradient Boosting machines.

The tool used in the study is **KNIME** Analytics platform (a tool to produce machine learning workflows). Google Colab for **python** coding is also used to produce results by coding and to compare results with that of KNIME.

THE FRUSTATING PROBLEM

While many organizations attempt to market their products to the consumer, it either ends as a frustration or as an item in the bin. If we are marketing a product to a client, we need to market the right product and, equally importantly, time the marketing campaign well in order to ensure that the consumer is able to subscribe/buy the product/offer. This is being designed to provide the "**Next Best Offer / Product**" as a service.

Thus, the key design elements that need to be considered are:
- **Data Sufficiency**
- **Target users/systems of the NBO/NBP engine**
- **Dimensions for assessment**
- **Feedback verification**

Using the existing dataset having customer demographics and other details, we need to predict **if the user will buy a term deposit or not**. More significantly, we should **recommend the term deposit to a customer or not**.

SAMPLE DATA SOURCE USED TO PREDICT PROBABILITY FOR BUYING A TERM DEPOSIT

- Bank client data [1]
  - Age (numeric)
  - Job Type : categorical:'admin.' ,'blue-collar', 'entrepreneur', 'housemaid', 'management','retired',self-employed','services','student','technician','unemployed','unknown')
  - Marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
  - Education (categorical: 'primary', 'secondary', 'tertiary', 'unknown')
  - default: has credit in default? (categorical: 'no','yes','unknown')
  - housing: has a housing loan? (categorical: 'no','yes','unknown')
  - loan: has a personal loan? (categorical: 'no','yes','unknown')

- Recent Contact Information:
  - contact: contact communication type (categorical: 'cellular','telephone')
  - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
  - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
  - 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.
- Other Attributes
  - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
  - 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)
  - previous: number of contacts performed before this campaign and for this client (numeric)
  - Poutcome: Outcome of previous marketing campaign (categorical: failure', 'nonexistent', 'success')
- Output variable **(desired target):**
  - y - has the client subscribed to a term deposit? (binary: 'yes','no')

Data Source: https://archive.ics.uci.edu/ml/datasets/bank+marketing

THE RIGHT ALGORITHM

While there are many techniques, we have arrived at xgboost as the algorithm used to predict the next best action basis the domain data sets. The key reasons for this are:

- **Availability as an open source in multiple forms**
- **Ease of training / re-training with feedback interchange for successive trainings**

From a coding perspective, we have used the KNIME analytics platform to implement the entire ML FLOW. This allows us to focus more on the key input dimensions that are required for predicting the possible next best action rather than building the complex xgboost into a package. This, using Knime means assembling a functionality that requires a heavy ML interlude.

THE OUTCOME, ITS INTERPRETATION AND WHAT NEXT

Here is a quick representation of the initial tests we conducted on our test data set. We observe that the outcome has yielded a RIGHT prediction on ½ cases.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y	Prediction (y)
2	42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5	may	380	1	-1	0	unknown	no	no
3	60	retired	married	tertiary	no	100	no	no	unknown	5	may	528	1	-1	0	unknown	no	yes
4																		
5																		
6																		
7																		

This implies that the data-set we had provided as input has either some bias or an incorrect classification or a hybrid. While most algorithms guarantee accurate predictions, the practical conclusion is that the input data-sets along with the classification columns are the key towards an accurate prediction. The model validation (using confusion matrix) within KNIME helps us discover the readiness and sufficiency of the data-sets:

Results

Confusion Matrix

Rows Number : 9043	no (Predicted)	yes (Predicted)	
no (Actual)	7719	245	96.92%
yes (Actual)	632	447	41.43%
	92.43%	64.60%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
no	7719	632	447	245	96.92%	92.43%	96.92%	41.43%	94.62%
yes	447	245	7719	632	41.43%	64.60%	41.43%	96.92%	50.48%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
90.30%	9.70%	0.454	8166	877

The classification machine learning model produced an overall accuracy score of around **90%** on an open-source banking dataset using the XGBoost algorithm. Classification reports suggest that target variable(y) was predicted with higher accuracy for 0(persons not having term deposits) than for 1(persons having term deposits). This is due to the **data imbalance/bias** as discussed above. Count of target (y) i.e. 0 and 1 in the dataset is 39922 and 5289 respectively. We can say that the data was biased towards the people who have not purchased the term deposits. So, in order to make much more accurate predictions, we would require a huge set of data with relevant features. Moreover, relevant pre-processing statistical techniques can be used to prepare the data before feeding it into the model.

SCALE AND CONTEXTUALITY

Machine learning/AI powered recommendation systems not only help banks identify the **relevant** customers who have a **high probability** of buying/subscribing to the products/offers but also in solving a larger problem - Scale. Banks or for that matter, any financial institution can deploy automated cross-sell and up-sell campaigns to their entire database and let the machine learn from its accuracy and drive customer contextuality in every action. Integrating the recommendation engine with Core Banking and CRM can further widen the learning curve.

APPENDIX

ABOUT XGBOOST ALGORITHM [2]

XGBoost is an open source library providing a high-performance implementation of gradient boosted decision trees. There are predominantly two categories in Ensemble techniques of machine learning namely: Bagging and Boosting. The XGBoost comes under the Boosting Ensemble learning. With a regular machine learning model, like a decision tree, a single model is trained on the dataset and is used for prediction. Even if we build a Random Forest (Bagging technique), all of the models are trained and applied to our data separately.

Boosting, on the other hand, takes a more iterative approach. It's still technically an ensemble technique but here many models are combined together to perform the final one, but takes a more clever approach.

Rather than training all of the models in isolation of one another, boosting trains models in succession, with each new model being trained to correct the errors made by the previous ones. Models are added sequentially until no further improvements can be made.

The advantage of this iterative approach is that the new models being added are focused on correcting the mistakes which were caused by other models.

KNIME MODEL [3]

KNIME Analytics Platform is the open source software for creating data science. KNIME makes understanding data easier and designing data science workflows in an ordered manner. The Node repository provides nodes for each and every step in the workflow. The complex data pre-processing, data visualization and machine learning technique can be modelled using the drag and drop graphical interface of Knime, without the need of coding. The model can then be deployed on the knime server which can further create REST API automatically.

WORKFLOW

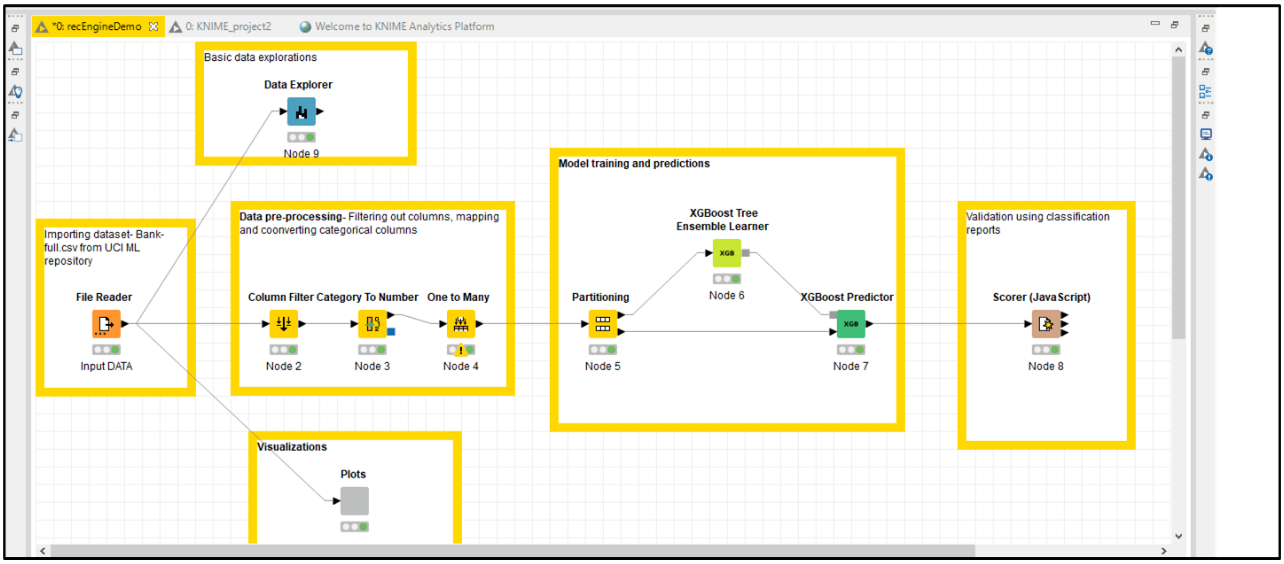


Fig 1: Knime Work

Nodes used in our study:

- Data Pre-processing
  File reader- To read the dataset locally stored in the pc
  Column filter- To filter out the columns which are not of our interest
  Category to Number- To map the categories into numbers
  One to many- To convert categorical variables into numerical ones

- Model training and predictions
  Partitioning- To split the data into training and test set (80:20)
  XGBoost ensemble learner- Training the model on training set
  XGBoost predictor- Predicting the output by passing test set as a node

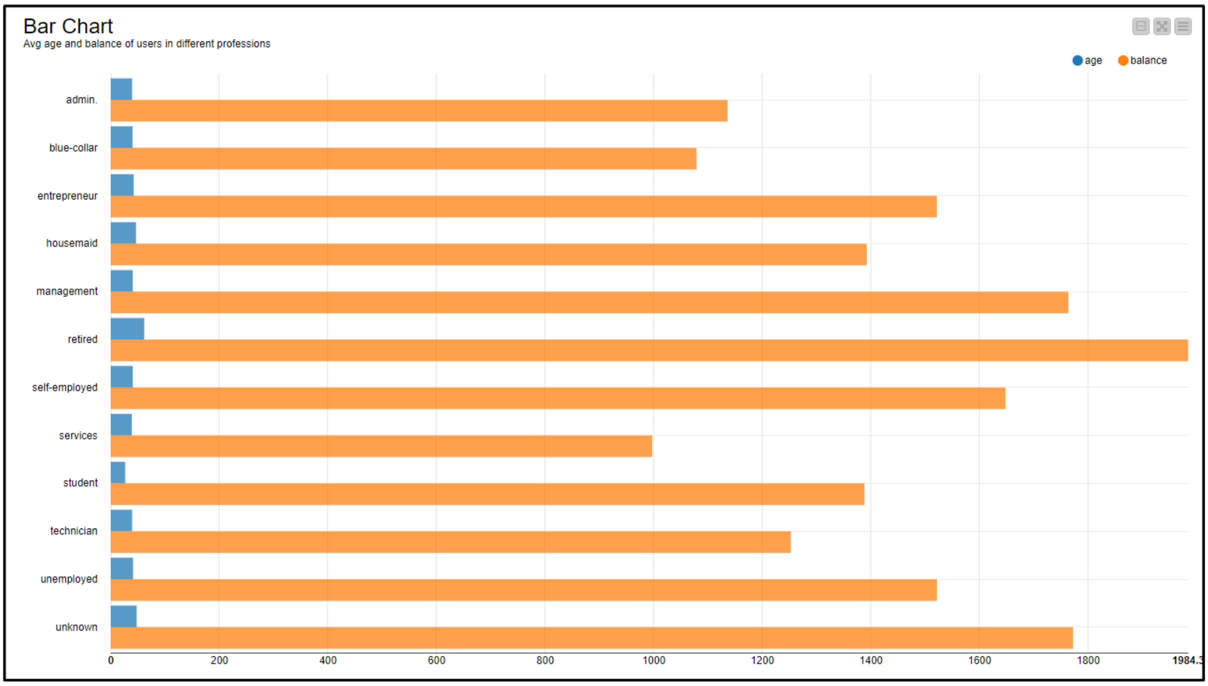
- Validation
  Scorer node- To validate our model using various parameters namely:
  Classification scores, Confusion matrix, Accuracy statistics etc.

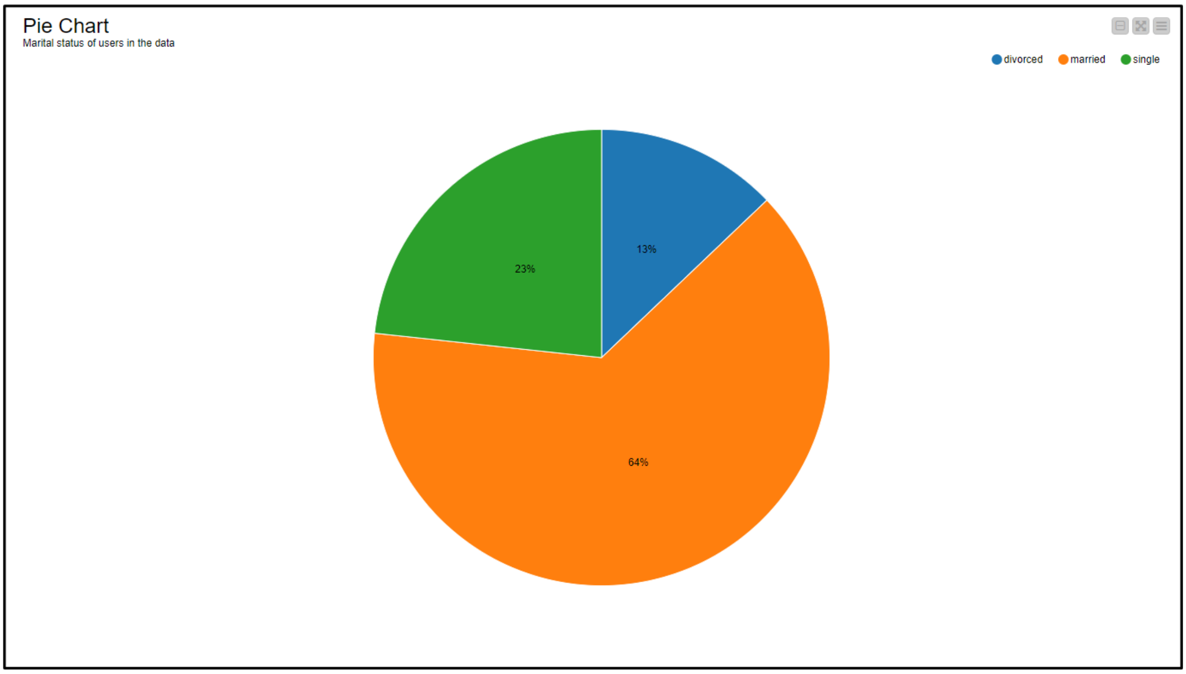
- Visualization nodes- Bar charts, histogram,pie charts etc.

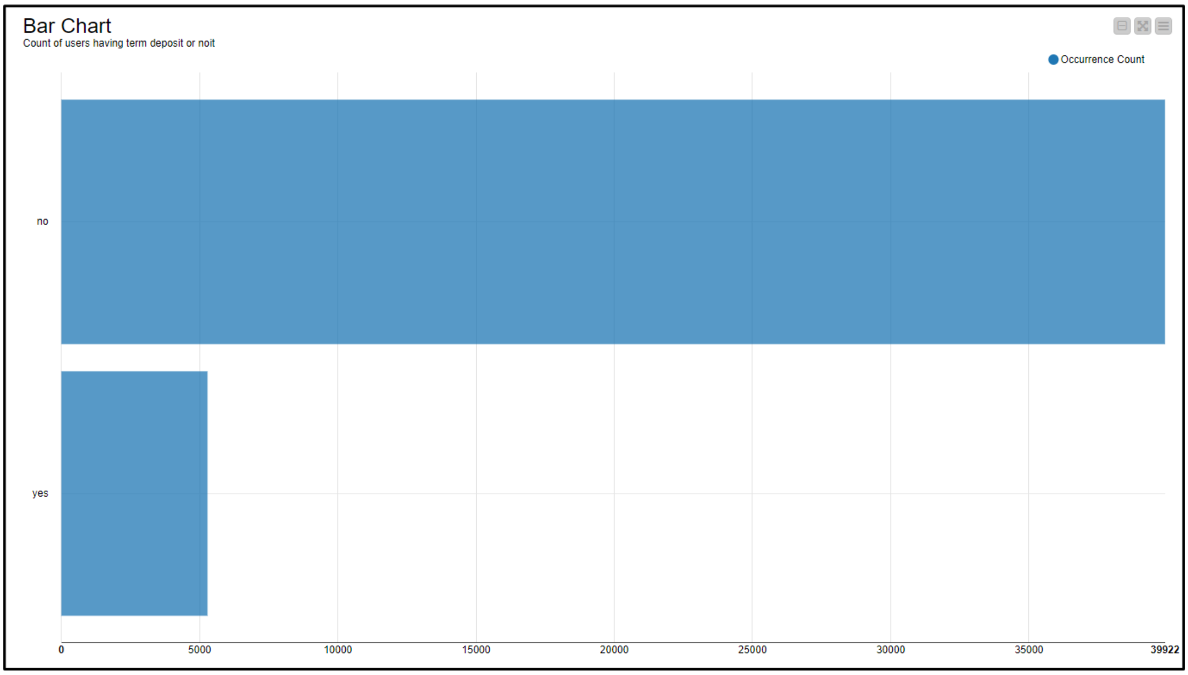
VISUALIZATIONS

Basic data visualizations were done using various plot nodes from the javascript library. This includes bar charts, pie charts etc.

All the visualization nodes were compiled into a single component named 'nodes'. This looks aesthetically good in a knime workflow. All the visualizations can be seen by using the interactive view option from the component node.







USING PYTHON

In our study, Google Colab was used for model training with Python using XGBoost. This creates a python notebook file (.ipynb) for our study.

The pandas library was used for data importing and pre-processing. The data was imported using the read_csv() method of pandas. The basic data preprocessing (checking duplicates, checking/dropping null rows, checking unique values, dropping irrelevant data etc) was done using suitable methods. The dataset used has 17 columns and the last column “y” is the target column which tells if a person has term deposit or not.

```
[ ] df=pd.read_csv("/content/drive/MyDrive/bank-full.csv", sep=";")
    df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

```
[ ] df.describe()
```

Fig 3: Importing Dataset

Map function was used to convert the yes/no categories column into 0/1 numeric columns in order to make it ready to use by the XGboost model.

```
df["loan"]= df["loan"].map({"yes": 1 , "no" : 0})
df["housing"]= df["housing"].map({"yes": 1 , "no" : 0})
df["default"]= df["default"].map({"yes": 1 , "no" : 0})
df["y"]= df["y"].map({"yes": 1 , "no" : 0})
```

```
[ ] df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	0	2143	1	0	unknown	5	may	261	1	-1	0	unknown	0
1	44	technician	single	secondary	0	29	1	0	unknown	5	may	151	1	-1	0	unknown	0
2	33	entrepreneur	married	secondary	0	2	1	1	unknown	5	may	76	1	-1	0	unknown	0
3	47	blue-collar	married	unknown	0	1506	1	0	unknown	5	may	92	1	-1	0	unknown	0
4	33	unknown	single	unknown	0	1	0	0	unknown	5	may	198	1	-1	0	unknown	0

Fig 4: Mapping

The categorical columns with “object” data type were converted into numerical columns using the **get_dummies()** method of pandas because the model can’t process the categorical columns.

Finally we used the **train_test_split()** method of sklearn library to split our data into training and testing sets. Test size was kept as 0.2 into the parameters. The XGBoost model instance was called and the classifier was instantiated. It was trained on the training data using **.fit()** method. Finally the predictions were made using the testing data using **.predict()** method.

```
import xgboost
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:235: D
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:268: D
  y = column_or_1d(y, warn=True)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Fig 5: Importing XGBoost

Accuracy metrics and classification reports were generated for the model using appropriate methods from **sklearn.metrics.**

```
from sklearn.metrics import classification_report, accuracy_score
accuracy_score(y_test, ypreds)
```

```
0.899037929890523
```

```
print(classification_report(y_test, ypreds))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      7952
           1       0.64      0.36      0.46      1091

    accuracy                           0.90      9043
   macro avg       0.78      0.67      0.70      9043
weighted avg       0.88      0.90      0.89      9043
```

Fig 6: Classification report

The confusion matrix and the accuracy scores in python were found to be giving almost similar results as produced by the KNIME workflow. The overall accuracy in python workflow came out to be 89.90% whereas in KNIME, it was 90.30%. A small difference might be due to the libraries that were used in the python workflow.

References

1. [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014
2. https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7
3. https://www.knime.com/knime-analytics-platform

Author:

Anurag Pal
Consultant, | i-GCB
Intellect Design Arena Limited

About Intellect Design Arena Limited

Intellect Design Arena Ltd. has the world's largest cloud-native, API-led microservices-based multi-product FinTech platform for global leaders in Banking, Insurance and Capital Markets. It offers a full spectrum of banking and insurance technology products through its four lines of businesses – Global Consumer Banking, Global Transaction Banking (iGTB), Risk, Treasury and Markets, and Insurance. With over 25 years of deep domain expertise, Intellect is the brand that progressive financial institutions rely on for their digital transformation initiatives.

Intellect pioneered Design Thinking to create cutting-edge products and solutions for banking and insurance, with design being the company's key differentiator in enabling digital transformation. FinTech 8012, the world's first design center for financial technology, reflects Intellect's commitment to continuous and impactful innovation, addressing the growing need for digital transformation. Intellect serves over 260 customers through offices in 97 countries and with a diverse workforce of solution architects and domain and technology experts in major global financial hubs around the world. For further information on the organization and its solutions, please visit **www.intellectdesign.com.**

Know more about IDC: https://www.igcb.com/digital-core.html
Contact: IGCB@intellectdesign.com