Leveraging CRISP-DM and PyCaret for Enhanced Client Engagement in Banking

Brahma Teja Chilumula
teja.btc07@gmail.com

Abstract

In the dynamic realm of banking, understanding client decision-making processes, especially concerning term deposits, is of paramount importance. This research delves into the intricacies of these decisions using the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology. Commencing with a keen understanding of the business context, the study systematically progresses through data preparation, modeling, and rigorous evaluation phases. Utilizing the PyCaret environment, a diverse range of models were compared, leading to the identification of significant influencing factors. Parallelly, an architectural overview of a microservices-based system was provided, offering insights into scalable system design. The findings present invaluable insights for banks aiming to optimize their marketing strategies, ultimately enhancing subscription rates and fostering stronger client relationships.

1 Introduction

- The banking sector continually seeks to optimize its marketing strategies to ensure it reaches potential clients effectively. The decision-making process of clients, particularly regarding term deposits, can be intricate and influenced by numerous factors. This research paper delves into these factors using the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology. Through a systematic approach, we aim to uncover the underlying variables that play a pivotal role in a client's decision.

2 Business Understanding

At the core of every data analysis lies the necessity to understand the business context. In the realm of banking, the subscription to term deposits represents a significant commitment from clients. For banks, understanding the reasons behind such decisions can lead to optimized marketing strategies, tailored offerings, and improved client relationships. Through this analysis, the objective is to discern the multifaceted factors that persuade a bank's client to commit to a term deposit..

3. Data Understanding and Preparation

Before diving deep into data modeling, it's paramount to grasp the nuances of the dataset at hand. Our initial steps involved importing necessary libraries and loading the dataset, which offers insights into various client attributes and their subscription status. Data quality is crucial; hence, we examined the dataset for missing values and inconsistencies. Data preparation, which includes handling of missing values and potential outliers, forms the bedrock of reliable outcomes.

Firstly, we need to install PyCaret in the colab notebook

“pip install pycaret[full]”

let’s import the necessary libraries and load our dataset to get an initial understanding:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pycaret.classification import *
from google.colab import files
uploaded = files.upload()


# Load the dataset with specified delimiters and quote character
data = pd.read_csv('bank.csv', delimiter=';', quotechar='"')


data.tail()


data.isnull().sum()
```

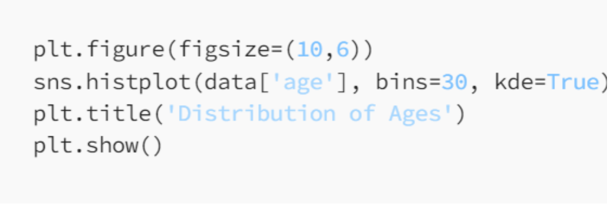
4. Data Modeling

Transitioning from data preparation, the modeling phase forms the crux of our analysis. The PyCaret environment, known for its efficiency and comprehensive suite of tools, was employed. By comparing a diverse range of models, we aimed to pinpoint those that resonate best with our dataset. Model selection isn't solely about accuracy; it's a blend of interpretability, performance, and alignment with business objectives. Our endeavors in this phase were geared towards finding that optimal balance.

4.1 Data Visualization

Visualization serves as a bridge between complex datasets and human understanding, translating intricate patterns into comprehensible insights. In this study, an intensive data visualization phase was undertaken post data preparation. Leveraging powerful visualization tools, we embarked on an exploratory journey to unearth hidden patterns, relationships, and anomalies in the dataset. From univariate distributions that offer a glimpse into individual attributes to multivariate plots that illuminate inter-variable relationships, each visual representation was meticulously crafted. These visuals not only illuminated the underlying structure of the data but also informed subsequent modeling decisions. By providing an intuitive lens into the dataset's landscape, this phase ensured that the subsequent analytical steps were rooted in a deep, visual understanding of the data's nuances.

some of the data visualization graphs:



```
plt.figure(figsize=(10,6))
sns.histplot(data['age'], bins=30, kde=True)
plt.title('Distribution of Ages')
plt.show()
```

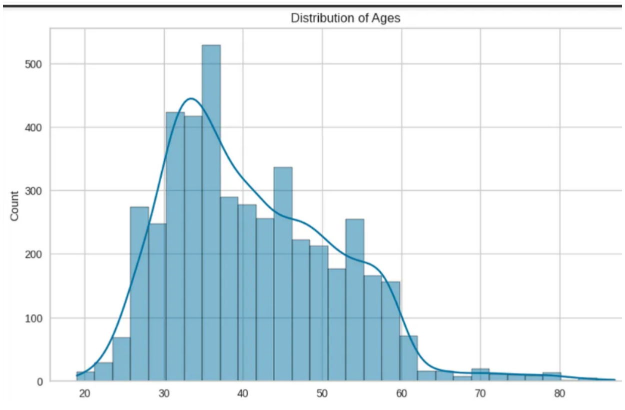
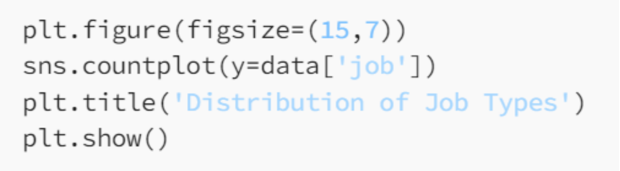


Fig 4 : Distribution of Ages graph.



```
plt.figure(figsize=(15,7))
sns.countplot(y=data['job'])
plt.title('Distribution of Job Types')
plt.show()
```

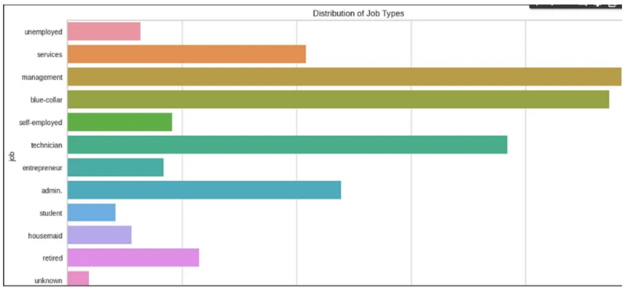
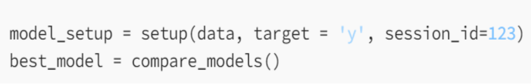


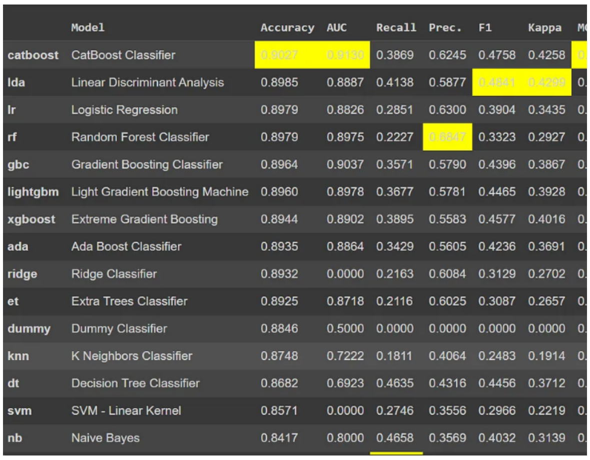
Fig 5 : Distribution of Job Types graph.

5. Evaluation

Modeling, while central to data analysis, is incomplete without rigorous evaluation. Post the training phase, we embarked on a journey to critically assess the performance of our chosen models. Key performance metrics were scrutinized to ensure the models not only fit the data well but also generalized effectively to unseen data. The culmination of this phase was the selection and saving of the best-performing model, ensuring reproducibility and ease of deployment in real-world scenarios.



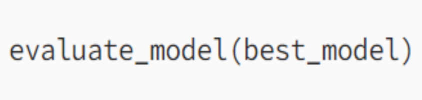
```
model_setup = setup(data, target = 'y', session_id=123)
best_model = compare_models()
```



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
catboost	CatBoost Classifier	0.9027	0.9130	0.3869	0.6245	0.4758	0.4258	0.
lda	Linear Discriminant Analysis	0.8985	0.8887	0.4138	0.5877	0.4841	0.4299	0.
lr	Logistic Regression	0.8979	0.8826	0.2851	0.6300	0.3904	0.3435	0.
rf	Random Forest Classifier	0.8979	0.8975	0.2227	0.6847	0.3323	0.2927	0.
gbc	Gradient Boosting Classifier	0.8964	0.9037	0.3571	0.5790	0.4396	0.3867	0.
lightgbm	Light Gradient Boosting Machine	0.8960	0.8978	0.3677	0.5781	0.4465	0.3928	0.
xgboost	Extreme Gradient Boosting	0.8944	0.8902	0.3895	0.5583	0.4577	0.4016	0.
ada	Ada Boost Classifier	0.8935	0.8864	0.3429	0.5605	0.4236	0.3691	0.
ridge	Ridge Classifier	0.8932	0.0000	0.2163	0.6084	0.3129	0.2702	0.
et	Extra Trees Classifier	0.8925	0.8718	0.2116	0.6025	0.3087	0.2657	0.
dummy	Dummy Classifier	0.8846	0.5000	0.0000	0.0000	0.0000	0.0000	0.
knn	K Neighbors Classifier	0.8748	0.7222	0.1811	0.4064	0.2483	0.1914	0.
dt	Decision Tree Classifier	0.8682	0.6923	0.4635	0.4316	0.4456	0.3712	0.
svm	SVM - Linear Kernel	0.8571	0.0000	0.2746	0.3556	0.2966	0.2219	0.
nb	Naive Bayes	0.8417	0.8000	0.4658	0.3569	0.4032	0.3139	0.

Fig : output for compare_models()

compare_models() function automatically runs the given dataset against the most of the different models and gives the metrics output of it , we can even arrange them based upon any of the criteria.



```
evaluate_model(best_model)
```

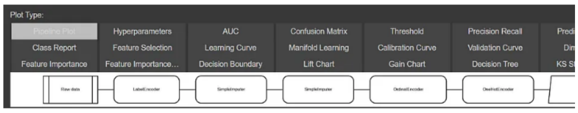
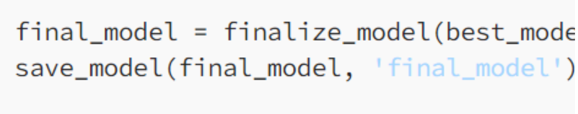
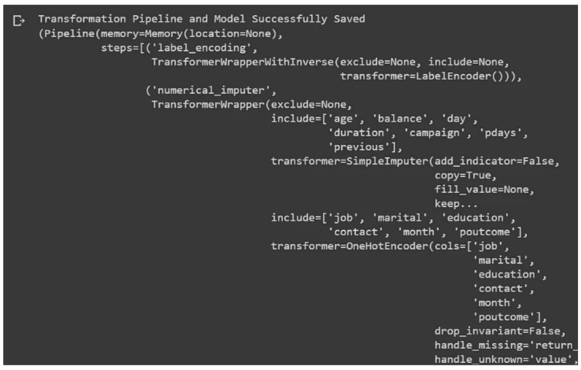


Fig 8 : evaluate_model() function output.

Saving the best model



```
final_model = finalize_model(best_mode
save_model(final_model, 'final_model')
```



```
Transformation Pipeline and Model Successfully Saved
(Pipeline(memory=Memory(location=None),
          steps=[('label_encoding',
                  TransformerWrapperWithInverse(exclude=None, include=None,
                                                transformer=LabelEncoder())),
                 ('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['age', 'balance', 'day',
                                              'duration', 'campaign', 'pdays',
                                              'previous'],
                                     transformer=SimpleImputer(add_indicator=False,
                                                               copy=True,
                                                               fill_value=None,
                                                               keep...
                                     include=['job', 'marital', 'education',
                                              'contact', 'month', 'poutcome'],
                                     transformer=OneHotEncoder(cols=['job',
                                                                     'marital',
                                                                     'education',
                                                                     'contact',
                                                                     'month',
                                                                     'poutcome'],
                                                               drop_invariant=False,
                                                               handle_missing='return_
                                                               handle_unknown='value',
```

Fig 10 : output for save_model() function.

7. Conclusion

This research journey, encapsulating the CRISP-DM methodology on a bank's dataset, has been enlightening. By meticulously traversing through each phase, from business understanding to model evaluation, we have unearthed insights that hold the potential to revolutionize marketing strategies in the banking sector.