**UCI Bank Marketing**

**Abstract**

The data is related with direct marketing campaigns of a banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

**Design**

In this Dataset from Kaggle we build a model to predict whether someone is going to make a deposit or not depending on some attributes. We will try to build 4 models using different algorithm Decision Tree, Random Forest, Naive Bayes, and K-Nearest Neighbors. After building each model we will evaluate them and compare which model are the best for our case.

**Data**

**Bank client data**

1. age (numeric)
2. job: type of job (categorical)
3. marital: marital status (categorical)
4. education (categorical)
5. default: has credit in default? (categorical)
6. housing: has housing loan? (categorical)
7. loan: has personal loan? (categorical)

**Related with the last contact of the current campaign**

1. contact: contact communication type (categorical)
2. month: last contact month of year (categorical)
3. day\_of\_week: last contact day of the week (categorical)
4. 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**

1. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
2. 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)
3. previous: number of contacts performed before this campaign and for this client (numeric)
4. poutcome: outcome of the previous marketing campaign (categorical)
5. Social and economic context attributes
6. emp.var.rate: employment variation rate - quarterly indicator (numeric)
7. cons.price.idx: consumer price index - monthly indicator (numeric)
8. cons.conf.idx: consumer confidence index - monthly indicator (numeric)
9. euribor3m: euribor 3-month rate - daily indicator (numeric)
10. nr. employed: number of employees - quarterly indicator (numeric)

**Output variable (target)**

* y - has the client subscribed a term deposit? (Binary: 'yes','no')

**Algorithms**

Feature Engineering

1. Converting categorical features to binary variables

2. Numerical transformations (scaling) after splitting

3. Category encoder (one-hot)

4. Apply SMOTE (Synthetic Minority Oversampling TEchnique) on data after spliting.

***Models***

will build 4 models using different algorithm Decision Tree, Random Forest, Naive Bayes, and K-Nearest Neighbours were used before settling on random forest as the model with strongest cross-validation performance. Random forest feature importance ranking was used directly to guide the choice and order of variables to be included as the model underwent refinement.

***Model Evaluation and Selection***

The entire training dataset of 45211 records was split into 80/20 train. and the model’s assessment is conducted using essential measurements, such as accuracy and recall, F1 and ROC are also used. the result for each method is:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classifier | Accuracy | Precision | recall | F1 | ROC |
| Decision Tree | 0.86 | 0.43 | 0.53 | 0.48 | 0.72 |
| Random Forest | 0.90 | 0.58 | 0.46 | 0.51 | 0.90 |
| Naive Bayes | 0.85 | 0.39 | 0.55 | 0.46 | 0.80 |
| K-Nearest Neighbours | 0.85 | 0.41 | 0.62 | 0.50 | 0.82 |

**Tools**

**Basic libs**

* import pandas as pd
* import numpy as np

**Data Visualization**

* import seaborn as sns
* import matplotlib.pyplot as plt

**Build the data Model**

* from sklearn.cluster import KMeans
* from sklearn import datasets
* from io import StringIO
* from sklearn.tree import export\_graphviz
* from sklearn import tree
* from sklearn import metrics
* %matplotlib inline
* from sklearn.linear\_model import LogisticRegression
* from sklearn.model\_selection import GridSearchCV
* from sklearn.neighbors import KNeighborsClassifier
* from sklearn.metrics import classification\_report, confusion\_matrix
* from sklearn.model\_selection import cross\_val\_score
* from sklearn.preprocessing import StandardScaler
* from sklearn.metrics import roc\_auc\_score,accuracy\_score,recall\_score
* from sklearn.metrics import roc\_curve
* from sklearn.model\_selection import train\_test\_split
* from sklearn.naive\_bayes import GaussianNB
* from sklearn.metrics import accuracy\_score
* from sklearn.svm import SVC
* from sklearn.tree import DecisionTreeClassifier
* from xgboost import XGBClassifier
* from sklearn.model\_selection import StratifiedKFold
* from sklearn.ensemble import RandomForestClassifier
* import warnings
* warnings.filterwarnings('ignore')
* from sklearn.preprocessing import MinMaxScaler
* from sklearn.preprocessing import PolynomialFeatures