Predicted Analysis

ALY6020 22757 - Module 6 Final Project

Final Project

Professor: Dr. Marco Montes de Oca

Northeastern University

2021 Winter CPS Quarter

Wan-Yi Lin [NUID: 001305931]

Liuzhao Tang [NUID: 001086160]

Date of Submission: 4/9/2021

Summary

The Objective of Our project is to study the historical marketing campaign data the bank performed, build a model to predict the clients' action (deposit or not), and identify the patterns that will help the bank find conclusions to develop future strategies.

Our goal is to create a model based on the previous marketing data to predict marketing activities based on historical records and find some features that can impact the activities performance.

Introduction

Our project is to study the historical marketing campaign data the bank performed, build a model to predict the clients' action (deposit or not), and identify the patterns that will help the bank find conclusions to develop future strategies.

The Bank Marketing Dataset comes from Kaggle.com, which contains 17 variables and a total of 11162 data. In addition, in this data set, there are categorical variables, including job, married, education, housing, loan, etc.; on the other hand, there are numeric variables, including age, balance, duration, etc. Table 1 shows the basic information of all columns.

Table 1

Columns information

Column	Description	Type
age	age of the customer	integer value
job	job of the customer	categorical feature
martial	marital status of the customer	categorical feature
education	education status	categorical feature
default	whether the customer is defaulter or not	categorical feature
balance	yearly account balance of the customer	continuous feature
housing	housing status of the customer	categorical feature
loan	whether the customer availed any loans	categorical feature
contact	how many times the customer has been contacted	categorical feature
day	day from last contact	discrete feature
month	month from last contacted date	categorical feature
duration	duration of last contact in hours	continuous feature
campaign	contact with how many campaign	categorical feature
pdays	the number of days since the previous campaign	numeric feature
previous	the number of contacts that reached the customer before this campaign	numeric feature
poutcame	previous campaign success	categorical feature
deposit	the customer deposit or not	Target variable, binary

Implementation

Considering the project problem is a classification problem, we plan to build several classification methods to predict clients' actions and choose the best one as our final model. Our workflow is as following:

Step 1: Pre-preparation

In this step, we install and load the library for future analysis. Also, to improve work effectiveness, we upload our dataset to Github.com and load the dataset from the web path instead of a local path to make sure anyone can run our code in any Internet environment.

```
# load modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost.sklearn import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
from sklearn.metrics import accuracy_score

# read data
url = 'https://media.githubusercontent.com/media/Carloszone/ALY-6020/master/Final%20proposal/bank.csv'

df = pd.read_csv(url)
```

Step 2: Data processing

First, we check the missing values in our dataset. For the missing values in numeric columns, we plan to use mean or median to replace them; for the missing values in

categorical columns, we can use mode to replace them or give a new label called “missing value”. Fortunately, our dataset’s data integrity is very well. No missing value has been checked.

```
# missing value check
missing = df.isna().sum().sort_values(ascending = False)
percent_missing = ((missing / df.isnull().count()) * 100).sort_values(ascending = False)
missing_df = pd.concat([missing,percent_missing], axis = 1, keys = ['Total', 'Percent'],sort = False)
missing_df[missing_df['Total'] >= 1]
```

Next, we do EDA to check the variable distribution and the relationship between response and other features. Then, we determine the following feature engineering strategies based on the EDA result.

```
# check target variable balance
sns.countplot(x = 'deposit', data =df)

#relationship between two categorical variables
sns.countplot(x = 'campaign', hue = 'deposit', data = df)
plt.xticks(rotation=60)
plt.legend(loc = 'upper right')
plt.tight_layout()

# relationship between two categorical variable and numeric variable
sns.violinplot(x = 'deposit', y = 'duration', data = df)
```

This code shows how we explore our dataset. For the target variable, deposit, we check the data balance first. Figure 1 is the barplot of the deposit, which implies the dataset is almost balanced. Figure 2 is an example to explore the relationship between two categorical variables. From figure 2, it is easy to find that only when campaign = 1, the “yes” group is greater than the “no” group. So, we can transform this variable into a binary variable to

reduce the feature numbers and improve moel speed. Figure 3 is the violin plot to check the relationship between a categorical variable and a numeric variable. The plot presents that the “no” group users’ duration is shorter than the “yes” group user.

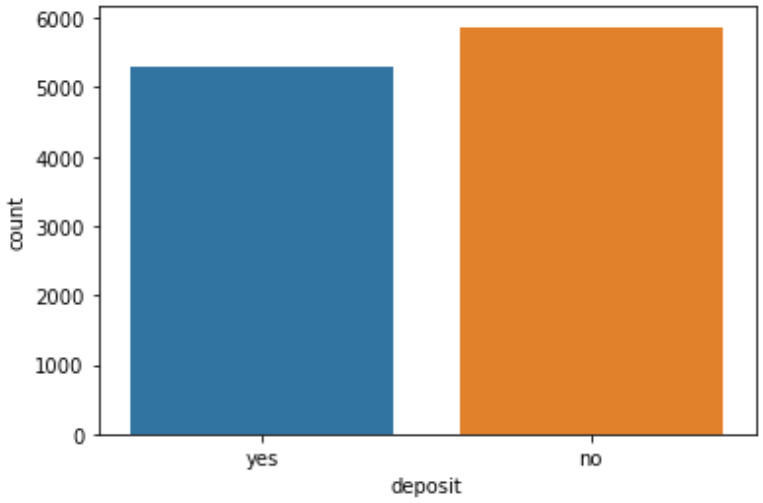

Figure 1: barplot of deposit

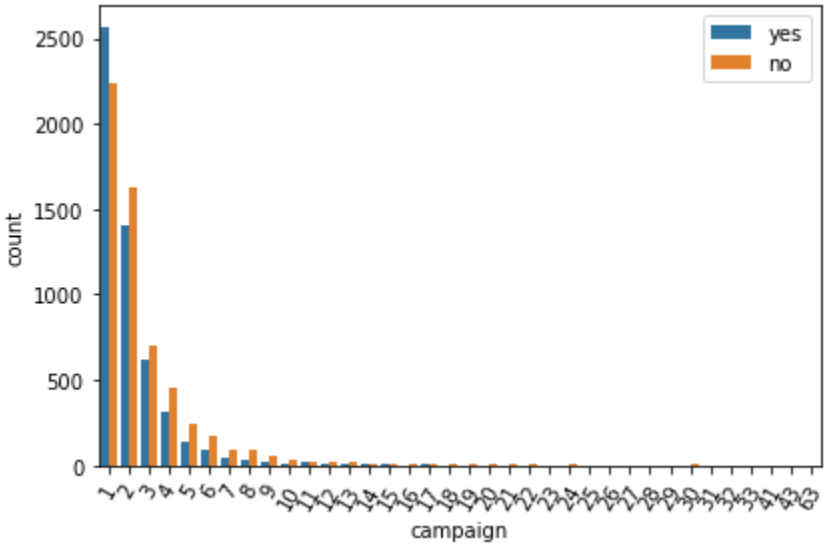

Figure 2: relationship between deposit and campaign

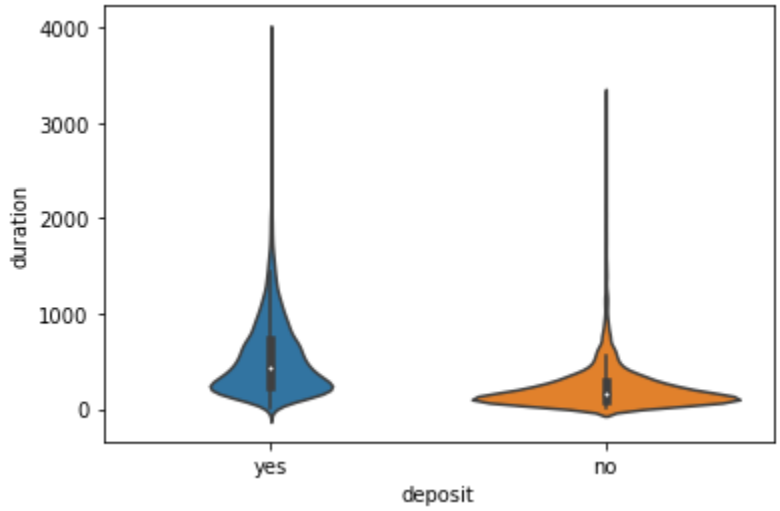

Figure 3: relationship between deposit and campaign

Step 3: Feature Engineering

Basically, we have two strategies to deal with features:

(1) Merge. If the column has many class values and part of the value has a similar distribution, we will merge these values and generate a new value.

(2) Transform. There are 3 situations to take this strategy:

First, for binary columns with “yes” and “no”, we will transform them to 0 and 1.

Second, for categorical variables, we plan to transform them to dummy variables

Third, some models will make decisions based on distance, so we must scale our features before we train our models. The code is as follows:

```
# feature engineering
Meger and transform variable to 0-1
df['campaign'] = df['campaign'].apply(lambda x: 1 if x == 1 else 0)
df['deposit'] = df['deposit'].apply(lambda x: 1 if x == 'yes' else 0)
df['default'] = df['default'].apply(lambda x: 1 if x == 'yes' else 0)
df['housing'] = df['housing'].apply(lambda x: 1 if x == 'yes' else 0)
df['loan'] = df['loan'].apply(lambda x: 1 if x == 'yes' else 0)

# generate dummy variables
df = pd.get_dummies(df,columns = ['job', 'marital', 'education', 'contact', 'day', 'month','poutcome'])
# relationship between two categorical variable and numeric variable
sns.violinplot(x = 'deposit', y = 'duration', data = df)

# data standardlization
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

Step 4: Models Selection

There are 4 models in our research plan: KNN, Logistic regression, Random forest, XGBoosting. Then, we pick up the best model as our final model. The detail of the models will be presented and discussed in the next part.

To get the best model performance, we should adjust some parameters for models. For the KNN model, I try to use different k parameters to find the best model parameter.

In logistic regression, we set “deposit” as the dependent variable to find out the relation of other variables. And we use the “binary” method to conduct logistic regression.

Next, in our ransom forest model, we set the number of trees to the default 500 trees. But we found that setting one hundred trees is enough for our dataset.

Training the XGBoosting model is more complex than training the rest of the models.

To find the best parameters, I fix the learning rate and iterator number and find the rest parameters by grid search. Then, I fix other parameters and find the best learning rate. Finally, I add the test set into the watchlist and check the model performance to find the best early stop parameter.

Step 5: Validation.

First, We split our dataset into the train set and test set. To compare the model performance of two sets, we can evaluate models and check the overfitting problem.

Second, in the grid search part, we implement 3 Fold cross-validation to improve speed. The grid search is used to find the best model hyperparameters. A higher cross-validation number can improve more robustness. However, this process is very time-consuming. So, we choose a litter CV number.

Third, we use accuracy as our model metric because our model is balanced.

```
# split dataset
X = df.drop('deposit',axis = 1)
Y = df['deposit']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2021)

# grid search
xgb_search = GridSearchCV(estimator = XGBClassifier(),
                          param_grid = param,
                          cv = 3,
                          n_jobs = -1,
                          verbose = 2)
```

Data Analysis

1. Logistic Regression

According to the results, the AIC value of this model is 7402.243. And the VIF value indicates that the result of "job" and "result" is greater than 4 but less than 5, and there may be slight collinearity.

```
> vif(bank_logit)
                GVIF Df GVIF^(1/(2*Df))
age         2.159318  1        1.469462
job         4.151633 11        1.066844
marital     1.444607  2        1.096320
education   2.349574  3        1.153006
default     1.021698  1        1.010791
balance     1.043313  1        1.021427
housing     1.393975  1        1.180667
loan        1.058674  1        1.028919
contact     1.846907  2        1.165766
day         1.287630  1        1.134738
month       3.186829 11        1.054095
duration    1.189442  1        1.090616
campaign    1.098730  1        1.048203
pdays       3.415335  1        1.848063
previous    1.592609  1        1.261986
poutcome    4.318695  3        1.276122
```

Figure4. VIF of logistic regression

```
> AIC(bank_logit)
[1] 7402.243
```

Figure5. AIC of logistic regression

In addition, there is a strong positive correlation between "deposit" and "monthly". Furthermore, there are some variables that have a strong positive correlation with deposits, including "students at school", "married single", "educational background", "number of days" and "duration", which means that when these When the value of the variable increases, people will have a higher probability of depositing. On the other hand, some variables have a strong negative correlation with deposits, including "loanyes", "contacttelephone", "housingyes", "monthmay", "monthaug", "monthnov" and "campaign", which means when People are less likely to save money when the values of these variables increase.

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-6.207e-01	2.987e-01	-2.078	0.037674	*
age	-2.600e-05	3.560e-03	-0.007	0.994173	
jobblue-collar	-3.692e-01	1.171e-01	-3.153	0.001614	**
jobentrepreneur	-4.524e-01	1.982e-01	-2.282	0.022466	*
jobhousemaid	-5.569e-01	2.182e-01	-2.552	0.010718	*
jobmanagement	-3.266e-01	1.204e-01	-2.713	0.006666	**
jobretired	3.016e-01	1.653e-01	1.825	0.068058	.
jobself-employed	-4.411e-01	1.763e-01	-2.502	0.012337	*
jobservices	-2.817e-01	1.341e-01	-2.101	0.035620	*
jobstudent	5.729e-01	1.971e-01	2.906	0.003656	**
jobtechnician	-1.902e-01	1.108e-01	-1.717	0.085971	.
jobunemployed	-1.680e-01	1.845e-01	-0.910	0.362740	
jobunknown	-5.573e-01	3.966e-01	-1.405	0.159925	
maritalmarried	-2.094e-01	9.628e-02	-2.175	0.029660	*
maritalsingle	5.886e-02	1.102e-01	0.534	0.593190	
educationsecondary	1.538e-01	1.044e-01	1.473	0.140743	
educationtertiary	4.259e-01	1.224e-01	3.479	0.000504	***
educationunknown	2.912e-01	1.662e-01	1.752	0.079741	.
default	-1.368e-01	2.521e-01	-0.543	0.587389	
balance	2.713e-05	9.592e-06	2.828	0.004677	**
housing	-7.189e-01	6.945e-02	-10.352	< 2e-16	***
loan	-4.785e-01	9.364e-02	-5.110	3.22e-07	***
contacttelephone	-9.582e-02	1.212e-01	-0.791	0.429073	
contactunknown	-1.559e+00	1.079e-01	-14.443	< 2e-16	***
day	4.793e-03	3.959e-03	1.211	0.225957	
monthaug	-9.037e-01	1.244e-01	-7.267	3.68e-13	***
monthdec	1.340e+00	4.380e-01	3.060	0.002210	**
monthfeb	-2.055e-01	1.423e-01	-1.445	0.148552	
monthjan	-1.308e+00	1.870e-01	-6.995	2.66e-12	***
monthjul	-1.009e+00	1.255e-01	-8.042	8.85e-16	***
monthjun	2.870e-01	1.486e-01	1.931	0.053450	.
monthmar	2.106e+00	2.726e-01	7.725	1.12e-14	***
monthmay	-7.097e-01	1.197e-01	-5.927	3.09e-09	***
monthnov	-1.036e+00	1.348e-01	-7.684	1.54e-14	***
monthoct	1.012e+00	1.946e-01	5.201	1.98e-07	***
monthsep	9.441e-01	2.290e-01	4.122	3.75e-05	***
duration	5.424e-03	1.385e-04	39.175	< 2e-16	***
campaign	-8.238e-02	1.494e-02	-5.515	3.49e-08	***

Figure 6. Important variables to “deposit”

2. KNN Model

The KNN model’s performance depends on the parameter k. Figure 7 shows the model performance. With the n increasing, the trend of train accuracy is down then fluctuating; the trend of test accuracy is up then fluctuating. when n = 11, the model gets the best test accuracy (0.735). It means that using the nearest 11 neighbors to predict will get the best result. The best KNN model can predict 73.5% of users’ action in this market activities.

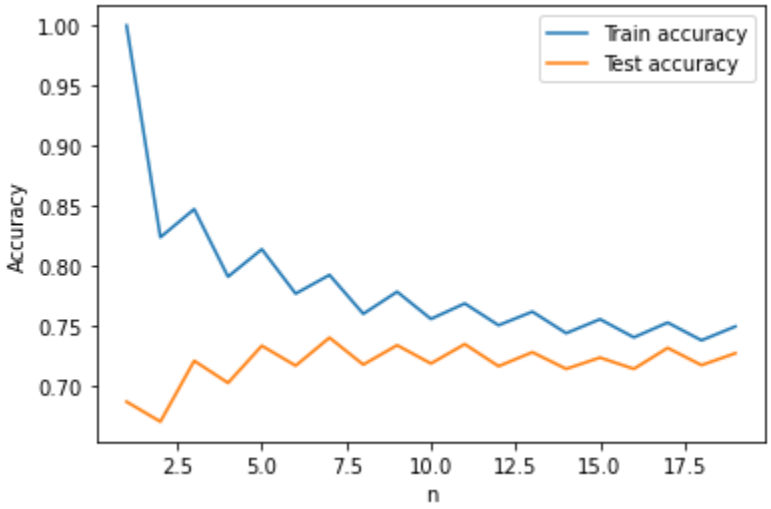

Figure 7: knn model result

The code of KNN model as follows. The code has two function: train and record the performance in test set, draw a plot about the test set accuracy.

```
def KNN(N):
    train_result = []
    test_result = []
    for n in range(1,N):
        model = KNeighborsClassifier(n_neighbors = n)
        model.fit(X_train,y_train)
        train_result.append(model.score(X_train,y_train))
        test_result.append(model.score(X_test, y_test))
        print('this is',n)
    return train_result, test_result

# record model performance
train_result, test_result = KNN(20)

# draw plot
n = list(range(1,20))
res = pd.DataFrame({'n':n,
                    'Train result':train_result,
                    'Test result': test_result})

sns.lineplot(data = res, x = 'n', y = 'Train result')
sns.lineplot(data = res, x = 'n', y = 'Test result')
plt.ylabel(ylabel = 'Accuracy')
plt.legend(loc = 'upper right', labels = ['Train accuracy', 'Test accuracy'])
```

3. XGBoosting Model

The test set accuracy of the XGBoosting model is 0.858, which means this model can predict almost 86% of users' actions. Figure 8 shows the feature importance. Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. According to figure 8, there is 4 features 'importance significantly higher than the rest of the features. They are balance, duration, age, and pdays.

```
model_xgb = xgb.train(final_param, dtrain,500, watchlist, early_stopping_rounds=100)
pred = model_xgb.predict(deval, ntree_limit=model_xgb.best_ntree_limit)
accuracy_score(y_test, pred)
```

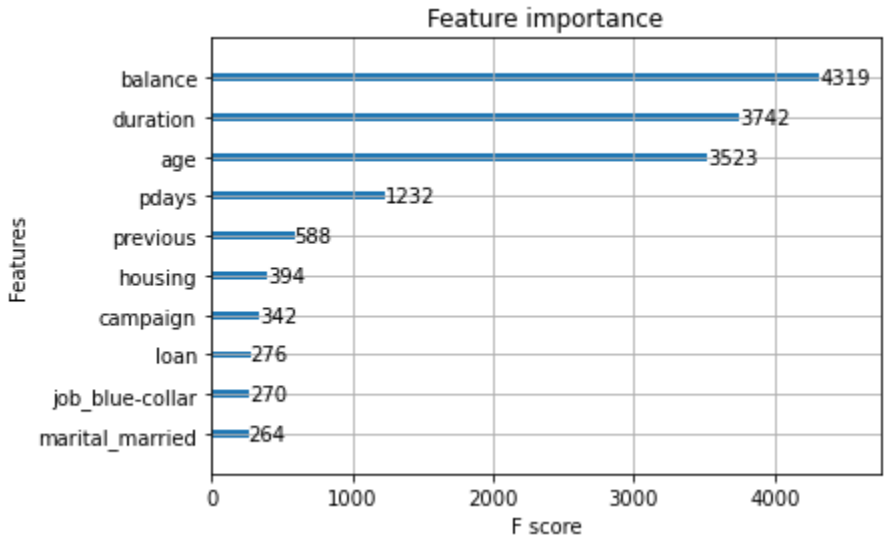

Figure 8: Feature importance in XGBoosting model (Top 10)

4. Random Forest Model

The accuracy of our random forest model is 0.8381, which is a model that can reach 83.81% of the predicted rate. In the figure, you can see the top three important variables are: duration, month, balance. After plotting the error rate graph calculated throughout of bag, we can see the first 20 trees are still at 20% error rate, and then as the forest is formed, it can be reduced to about 5% error rate.

```
rf <- randomForest(deposit~., data=train_data, importance=T, proximity=T, do.trace=100)
plot(rf)
round(importance(rf),2)
result <- predict(rf, newdata= test_data)
result_Approved <- ifelse(result > 0.5,1,0)
confusionMatrix(as.factor(result_Approved), as.factor(test_data$deposit))
```

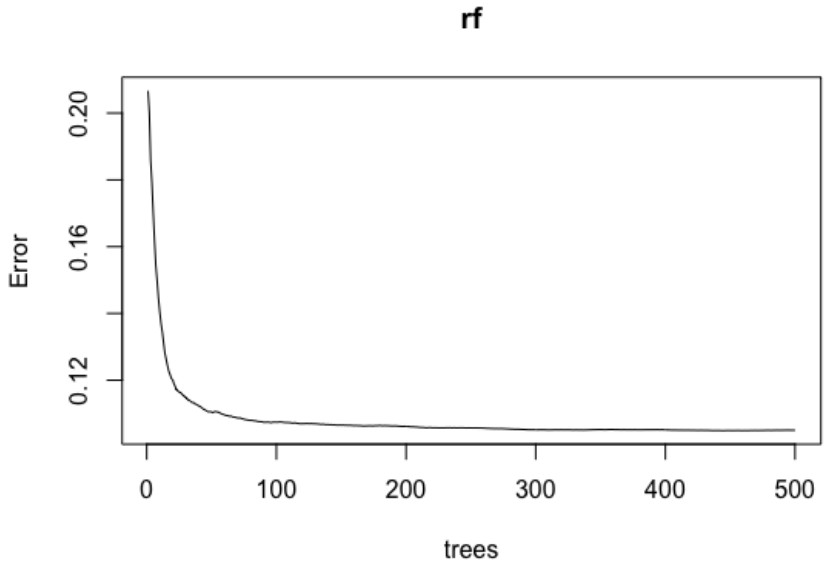

Figure 9. Random forest error rate in 0-500 trees

Conclusion

Table 3 presents the 4 models' performance. Based on table 3, we can conclude that the XGBoosting model is the best model, and its accuracy is 0.858.

Also, because balance, duration, age, and pdays are the most important features in the predictive model, the bank should design different strategies for different user types based on these features in the future market activities.

Table 3

Model performance result

Model	Accuracy
KNN	0.741
XGBoosting	0.852
Logistic Regression	0.8381
Random Forest	0.8381

Discussion

First, in business activities, one of analytic benefits is to reduce the cost of operation and improve the work effectiveness. So, learning from historical data and generating new marketing strategies are very profitable. That's why we are interested in this topic. Based on our result, we locate some important features, and it is useful for the bank to classify users group and take different marketing strategies to improve marketing performance.

In this research, comparing the 4 models' performance, we can find that the gradient descent algorithm is the best algorithm to optimize the model. This algorithm is based on gradient, and the gradient is based on the loss function. So, maybe we can use different loss functions to check the result, and there is the best loss function to help predict our target variable.

To improve the model performance, I think there are two options: (1) add more data or features; (2) try more models and algorithms. A bigger dataset can precisely reflect the population distribution, and training more models will improve the possibility to find a better result. So, if a model can learn from a bigger dataset, it may get better performance and offer more valuable insights. Also, for this case, I think the ensemble model is not a good choice. A weakness of the ensemble model is poor interpretability. Therefore, if we get a better result from an ensemble model, we cannot use the model to guide practical operation. The bank still don't know how to improve its marketing strategy.

References

Chauhan, G.(2018) All about Logistic regression in one article. Retrieved from: https://towardsdatascience.com/logistic-regression-b0af09cdb8ad

Data source: (2018) Bank Marketing Dataset https://www.kaggle.com/janiobachmann/bank-marketing-dataset

Sharma, A.(2020) Decision Tree vs. Random Forest – Which Algorithm Should you Use? Retrieved from: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

Yıldırım, S.(2020) K-Nearest Neighbors (kNN) — Explained. Retrieved from: https://towardsdatascience.com/k-nearest-neighbors-knn-explained-cbc31849a7e3