Banking Dataset - Marketing Targets

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df1 = pd.read_csv('train.csv',sep=';')
df2 = pd.read_csv('test.csv',sep=';')
```

```
df1.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day
0	58	management	married	tertiary	no	2143	yes	no	unknown	5
1	44	technician	single	secondary	no	29	yes	no	unknown	5
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5
4	33	unknown	single	unknown	no	1	no	no	unknown	5

```
df2.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day
0	30	unemployed	married	primary	no	1787	no	no	cellular	19
1	33	services	married	secondary	no	4789	yes	yes	cellular	11
2	35	management	single	tertiary	no	1350	yes	no	cellular	16
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5

```
df = pd.concat([df1, df2], ignore_index= True)
```

```
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day
0	58	management	married	tertiary	no	2143	yes	no	unknown	5
1	44	technician	single	secondary	no	29	yes	no	unknown	5
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5
4	33	unknown	single	unknown	no	1	no	no	unknown	5

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49732 entries, 0 to 49731
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        49732 non-null  int64 
 1   job        49732 non-null  object
 2   marital    49732 non-null  object
 3   education  49732 non-null  object
 4   default    49732 non-null  object
 5   balance    49732 non-null  int64 
 6   housing    49732 non-null  object
 7   loan       49732 non-null  object
 8   contact    49732 non-null  object
 9   day        49732 non-null  int64 
 10  month      49732 non-null  object
 11  duration   49732 non-null  int64 
 12  campaign   49732 non-null  int64 
 13  pdays      49732 non-null  int64 
 14  previous   49732 non-null  int64 
 15  poutcome   49732 non-null  object
 16  y          49732 non-null  object
dtypes: int64(7), object(10)
memory usage: 6.5+ MB
```

```
# Find null values in dataset
df.isnull().sum()
```

```
           0
      age  0
      job  0
  marital  0
education  0
  default  0
  balance  0
  housing  0
     loan  0
  contact  0
      day  0
    month  0
 duration  0
 campaign  0
    pdays  0
 previous  0
 poutcome  0
        y  0

dtype: int64
```

```
df.describe()
```

```
                age        balance           day      duration      campaign         pd
count  49732.000000   49732.000000  49732.000000  49732.000000  49732.000000  49732.000
 mean     40.957472    1367.761562     15.816315    258.690179      2.766549     40.158
  std     10.615008    3041.608766      8.315680    257.743149      3.099075    100.127
  min     18.000000   -8019.000000      1.000000      0.000000      1.000000     -1.000
  25%     33.000000      72.000000      8.000000    103.000000      1.000000     -1.000
  50%     39.000000     448.000000     16.000000    180.000000      2.000000     -1.000
  75%     48.000000    1431.000000     21.000000    320.000000      3.000000     -1.000
  max     95.000000  102127.000000     31.000000   4918.000000     63.000000    871.000
```

```
x = df.drop(['y'],axis = 1)
y =df.y
```

```
y.head()
```

```
   y
0  no
1  no
2  no
3  no
4  no

dtype: object
```

```
# Store all categorical (text) column into dataframe
categorical_columns = df.select_dtypes(include=['object']).columns
```

```
categorical_columns
```

```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'poutcome', 'y'],
      dtype='object')
```

```
#Import labelencoder for converting string to number.
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
#Converting Categorical columns in Numeric for training M.L. model
for col in categorical_columns:
  df[col]=le.fit_transform(df[col])
```

```
df.head()
```

```
   age  job  marital  education  default  balance  housing  loan  contact  day  month
0   58    4        1          2        0     2143        1     0        2    5      8
1   44    9        2          1        0       29        1     0        2    5      8
2   33    2        1          1        0        2        1     1        2    5      8
3   47    1        1          3        0     1506        1     0        2    5      8
4   33   11        2          3        0        1        0     0        2    5      8
```

```
#Define independent variable into x and dependent into y.

#Independents variables

x1= df.drop(['y'],axis=1)
x1.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	58	4	1	2	0	2143	1	0	2	5	8
1	44	9	2	1	0	29	1	0	2	5	8
2	33	2	1	1	0	2	1	1	2	5	8
3	47	1	1	3	0	1506	1	0	2	5	8
4	33	11	2	3	0	1	0	0	2	5	8

```
#Dependent variable
y1=df.y
y1.head()
```

	y
0	0
1	0
2	0
3	0
4	0

dtype: int64

```
#Find best parameters using hyper parameter tuning
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV


# Find the best parameters.
model_params = {

    'random_forest': {
        'model': RandomForestClassifier(),
        'params': {
            'n_estimators': [0,1, 5, 10]
        }
    }
}
```

```
scores = []

for model_name, mp in model_params.items():

    clf = GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=False)
    clf.fit(x1, y1)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })


df1 = pd.DataFrame(scores)
df1
```

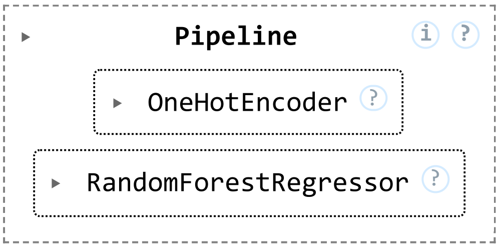
	model	best_score	best_params
0	random_forest	0.842273	{'n_estimators': 10}

```
# Create a Pipeline to Encode Categorical Features Numerically and Train a Model

from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import OneHotEncoder

# Define the pipeline
clf = Pipeline([
    ('encodef', OneHotEncoder()),  # Encoding categorical features
    ('mod', RandomForestRegressor(n_estimators=10))  # Random Forest model
])


clf.fit(x,y1)
```



```
clf.score(x,y1)
```

```
0.8738745372507875
```

Our model achieves an accuracy of 87%.

```
columns = ['age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan',
           'contact', 'day', 'month', 'duration', 'campaign', 'pdays', 'previous', 'poutc

new_data_points = [
    [59, 'admin.', 'married', 'secondary', 'no', 2343, 'yes', 'no', 'unknown', 5, 'may',
]

input = pd.DataFrame(new_data_points, columns=columns)


# Test the model based on above input.

prediction= clf.predict(input)[0]
```