Project - Classification

Bank Marketing

First analyse & data cleaning

- Dataset of direct marketing campaigns (phone calls) of a banking institution.

  → The classification goal is to predict if the client will subscribe a term deposit

- Dataset structure
  - 17 columns:
    - numerical (age, balance...)
    - categorical (job, education, marital...)
    - binary (housing loan, personal loan...)
  - 4521 rows

  - Removal of column 'duration' :
    - the duration is not known before a call is performed. Also, after the end of the call y is obviously known.

Input variables:

- *bank client data:*

1 - age	*(numeric)*
2 - job : type of job	*(categorical)*
3 - marital : marital status	*(categorical: "married","divorced","single")*
4 - education	*(categorical)*
5 - default: has credit in default?	*(binary: "yes","no")*
6 - balance: average yearly balance, in euros (numeric)	
7 - housing: has housing loan?	*(binary: "yes","no")*
8 - loan: has personal loan?	*(binary: "yes","no")*

- *related with the last contact of the current campaign:*

9 - contact: contact communication type	*(categorical)*
10 - day: last contact day of the month	*(numeric)*
11 - month: last contact month of year	
12 - duration: last contact duration, in seconds	*(numeric)*

- *other attributes:*

13 - campaign: number of contacts performed during this campaign and for this client (numeric)
14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
15 - previous: number of contacts performed before this campaign and for this client (numeric)
16 - poutcome: outcome of the previous marketing campaign (categorical)

Output variable (desired target):

17 - y - has the client subscribed a term deposit? (binary: "yes","no")

Classification algorithms

- Two different tests on categorical columns:
  1. with ‘get_dummies’
  2. with replacing the existing text with the new encoded data.

- **LogisticRegression is the best model in the 2 cases**
  - class_weight=’balanced’ really helps (only 11% of yes in the target column)
  - the most important parameter here is recall

1. Get_dummies

```
lr=LogisticRegression(max_iter=10000, class_weight='balanced')
lr.fit(X_train, y_train)
y_pred=lr.predict(X_test)

conf1=confusion_matrix(y_test, y_pred)
acc1=accuracy_score(y_test, y_pred)
rec1=recall_score(y_test, y_pred)
pre1=precision_score(y_test, y_pred)
f11=f1_score(y_test, y_pred)
display(conf1)
print('Accuracy:', acc1)
print('Recall:', rec1)
print('Precision:', pre1)
print('F1:', f11)

measures['LogisticRegression']=[acc1, rec1, pre1, f11]
```

```
array([[544, 219],
       [ 40,  60]], dtype=int64)

Accuracy: 0.6998841251448435
Recall: 0.6
Precision: 0.21505376344086022
F1: 0.31662269129287596
```

	Accuracy	Recall	Precision	F1
LogisticRegression	0.699884	0.60	0.215054	0.316623
GaussianNB	0.834299	0.41	0.328000	0.364444
Decision Tree Classifier	0.814600	0.30	0.250000	0.272727
AdaBoost	0.894554	0.20	0.645161	0.305344
CatBoost	0.894554	0.20	0.645161	0.305344
XGBoost	0.885284	0.19	0.513514	0.277372
Random Forest	0.882966	0.13	0.481481	0.204724
KNN	0.871379	0.12	0.342857	0.177778
Random Forest Balanced	0.885284	0.12	0.521739	0.195122
SVC	0.884125	0.00	0.000000	0.000000

2. Encoded categories

```
lr=LogisticRegression(max_iter=10000, class_weight='balanced')
lr.fit(X_train, y_train)
y_pred=lr.predict(X_test)
conf1=confusion_matrix(y_test, y_pred)
acc1=accuracy_score(y_test, y_pred)
rec1=recall_score(y_test, y_pred)
pre1=precision_score(y_test, y_pred)
f11=f1_score(y_test, y_pred)
display(conf1)
print('Accuracy', acc1)
print('Recall', rec1)
print('Precision', pre1)
print('F1', f11)

measures['LogisticRegression']=[acc1, rec1, pre1, f11]
```

```
array([[526, 237],
       [ 41,  59]], dtype=int64)

Accuracy 0.6778679026651216
Recall 0.59
Precision 0.19932432432432431
F1 0.297979797979796
```

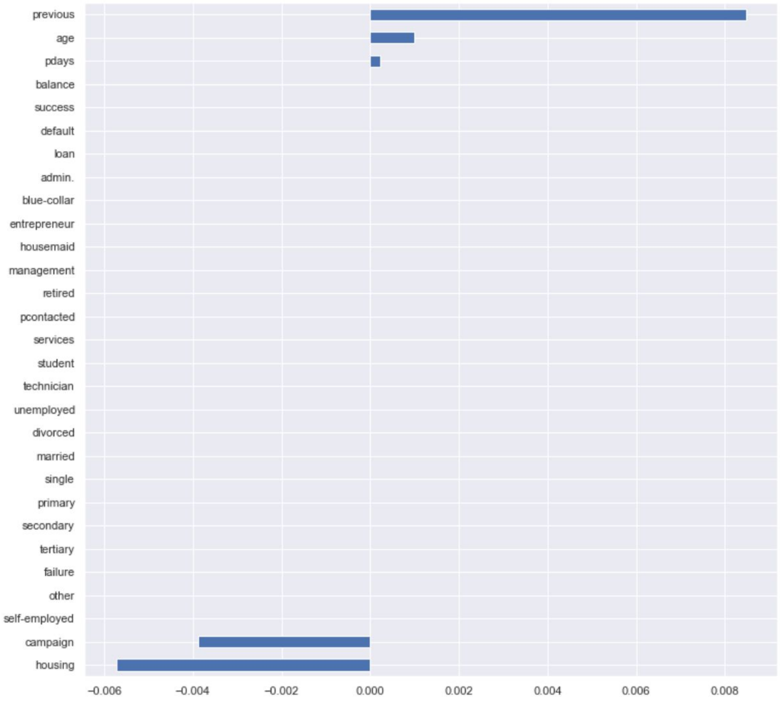
	Accuracy	Recall	Precision	F1
LogisticRegression	0.677868	0.59	0.199324	0.297980
GaussianNB	0.809965	0.31	0.246032	0.274336
Decision Tree Classifier	0.807648	0.30	0.238095	0.265487
XGBoost	0.882966	0.20	0.487805	0.283688
AdaBoost	0.891078	0.18	0.600000	0.276923
CatBoost	0.891078	0.18	0.600000	0.276923
Random Forest	0.887601	0.14	0.560000	0.224000
Random Forest Balanced	0.885284	0.11	0.523810	0.181818
KNN	0.872538	0.10	0.333333	0.153846
SVC	0.884125	0.00	0.000000	0.000000

Features selection

- Tests with different methods::
  - Best recall with Lasso
  - these columns have big impacts on the target:
    - previous
    - age
    - campaign
    - housing

	Accuracy	Recall	Precision	F1
LogisticRegression_LassoCV	0.581692	0.67	0.169620	0.270707
LogisticRegression	0.699884	0.60	0.215054	0.316623
LogisticRegression_RidgeCV	0.713789	0.57	0.218391	0.315789
LogisticRegression_RFE	0.754345	0.53	0.243119	0.333333
LogisticRegression_SFS	0.860950	0.18	0.321429	0.230769

Lasso



To conclude

After training our data, and testing it, we can conclude that the Logistic Regression using feature selection with Lasso is the best model to predict if a client will subscribe a term deposit or not.

Difficulties / Learnings / Improvements

- Difficulties :
  - imbalanced classification

- Learnings :
  - use features selection & classification models
  - use class_weight='balanced'

- Improvements:
  - use SMOTE for imbalanced classification :
    - SMOTE will alter the data and make the dataset balanced by oversampling (means it will generate similar looking data as in minority class to increase its samples).
  - https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Thanks!

Does anyone have any questions?