IT 461 Practical Machine Learning

Fall 2020

Project Part II

Group Member Names:

Arwa Alqahaiz	438200815
Sarah Ali Bajaba	438202742

Main Topic of Problem: predict if the client will subscribe a term deposit after a marketing campaigns of a Portuguese banking institution.

I. Introduction

a) Introducing the topic and why is it useful to use ML algorithms on it.

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y). The marketing campaigns were based on phone calls. Often, more than one contact to the same client is required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed which means that the binary classification productive modules are the most suitable modules for this problem. We choose the logistic regression and SVM model to train the data with and predict the result as two classes that will let us know which columns (variables) has the most effect on whether the client will subscribe a term deposit after a marketing campaigns of banking institution.

we have 421188 observations and 21 features.

- **Features:**
  - Age: (numeric)
  - Job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
  - Marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown' ; note: 'divorced' means divorced or widowed)
  - Education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
  - Default: has credit in default? (categorical: 'no', 'yes', 'unknown')
  - Housing: has housing loan? (categorical: 'no', 'yes', 'unknown')
  - Loan: has personal loan? (categorical: 'no', 'yes', 'unknown')
  - Contact: contact communication type (categorical: 'cellular','telephone')

- Month: last contact month of year (categorical: 'jan', 'feb', 'mar', …, 'nov', 'dec')
- Day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
- Duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no').
- Campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- 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)
- Previous: number of contacts performed before this campaign and for this client (numeric)
- Poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
- Emp.var.rate: employment variation rate - quarterly indicator (numeric)
- Cons.price.idx: consumer price index - monthly indicator (numeric)
- Cons.conf.idx: consumer confidence index - monthly indicator (numeric)
- Euribor3m: euribor 3 month rate - daily indicator (numeric)
- Nr.employed: number of employees - quarterly indicator (numeric)
- y :has the client subscribed a term deposit? (binary: 'yes', 'no')

b) What is the problem you want to solve and what are you proposing to solve it with?

Since the dataset is for marketing campaigns of banking institution and whether it success or not; The classification goal is to predict if the client will subscribe a term deposit (variable y) = the marketing campaign is successful or unsuccessful. And that will help the bank to focus more on the most effective factors. The marketing campaigns were based on phone calls. Often, more than one contact to the same client is required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed which means that the binary classification productive modules are the most suitable modules for this problem.

II. Data Preprocessing

1- Show what preprocessing techniques you will be using

We did some pre-processing techniques on the data to reduces complexity of the data under analysis as data in real world is unclean.

We check if the data has null values to drop it, and dropped the duplicated rows in the data since they are few also we balance the 'Y' column and check the outliers of age column, finally we transform the columns that we will use to apply the algorithms on, like: Loan and Housing

2- Why did you choose to apply these techniques?

The biggest advantage of pre-processing in ML is to improve generalizability of your model. Also, to prepare the data so that when using prepared data, the methods will produce better models, faster result

The dataset has a few unknown values so we decided to drop it since it won't affect the result, and we transformed some columns so we can apply the ML algorithms on it

also we balance the data because we have a serious amount of bias on 'y' column so we adjust it to make the result more efficient and reasonable.

Provide examples of preprocessing steps

1- First we check of the data has any null values to handle it but there wasn't any null

Ckeck the null value

```
In [9]: df.isnull().sum()

Out[9]: age               0
        job               0
        marital           0
        education         0
        default           0
        housing           0
        loan              0
        contact           0
        month             0
        day_of_week       0
        duration          0
        campaign          0
        pdays             0
        previous          0
        poutcome          0
        emp.var.rate      0
        cons.price.idx    0
        cons.conf.idx     0
        euribor3m         0
        nr.employed       0
        y                 0
        dtype: int64
```

2- Check the duplicates on the dataset

Check for duplicates in the dataset

```
len(df)-len(df.drop_duplicates())

]: 12
```

3- Then drop the duplicates

Drop the duplicates in the dataset

```
df = df.drop_duplicates()
```

```
df.shape
```

```
(41176, 21)
```

4- Transform the ‘y’ values into 0,1 to apply the methods on it

```
df['y'].value_counts()
```

```
no     36537
yes     4639
Name: y, dtype: int64
```

```
def trans(x):
    if x=='no' :
        return 0
    elif x=='yes' :
        return 1
```

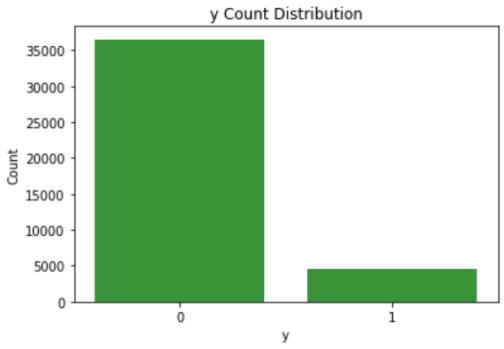
```
df.y.replace(('yes', 'no'), (1, 0), inplace=True)
```

5- Balance the ‘y’ column

the Y columns before we do the balance

```
uniform_color=sns.color_palette()[2]
sns.countplot(data=df,x='y',color=uniform_color)
plt.xlabel('y')
plt.ylabel('Count')
plt.title('y Count Distribution')
```

```
Text(0.5, 1.0, 'y Count Distribution')
```




```
# example of random undersampling to balance the class distribution
from collections import Counter
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.under_sampling import RandomUnderSampler
# define dataset
X = df[['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y']]
y = df['y']
# summarize class distribution
print()
print("The data distrubtion before undersampling :")
print(Counter(y))
print()
# define undersample strategy
undersample = RandomUnderSampler(sampling_strategy='majority')
# fit and apply the transform
X_over, y_over = undersample.fit_resample(X ,y)
# summarize class distribution
print("The data distrubtion after undersampling :")
print(Counter(y_over))
print()
```

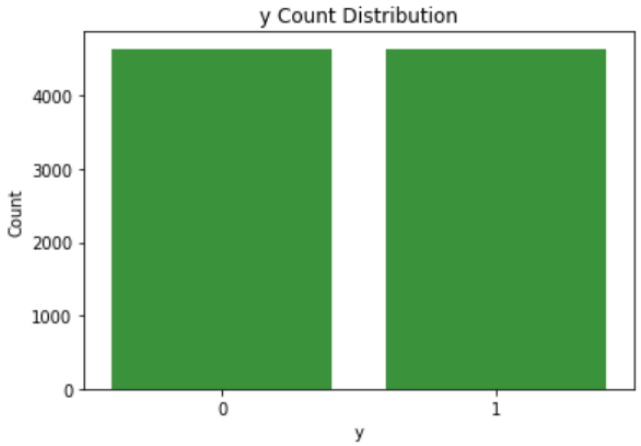
The result:

```
The data distrubtion before undersampling :
Counter({0: 36537, 1: 4639})

The data distrubtion after undersampling :
Counter({0: 4639, 1: 4639})
```

the Y columns after we do the balance

```
uniform_color=sns.color_palette()[2]
sns.countplot(data=df,x='y',color=uniform_color)
plt.xlabel('y')
plt.ylabel('Count')
plt.title('y Count Distribution')
```

1]: `Text(0.5, 1.0, 'y Count Distribution')`

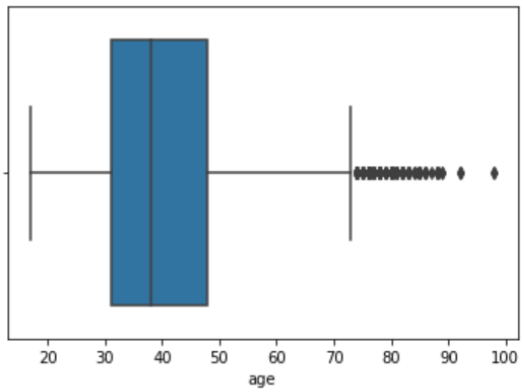


6- Outliers: evaluate the outliers on age column

Lets evaluate the outlier

```
print(df['age'].max())
print(df['age'].min())
sns.boxplot(df['age']);
```

```
98
17
```



Age has some outliers

7- Drop columns: because it has the same value so it won't affect the result

```
df=df.drop(columns=['default'],axis=1)
```

8- Transform 'housing' column to numerical and drop unknown values since it's few

Housing: has housing loan? (categorical: 'no', 'yes', 'unknown')

```
df['housing'].value_counts()
```

```
']: yes        4873
    no         4174
    unknown     231
    Name: housing, dtype: int64
```

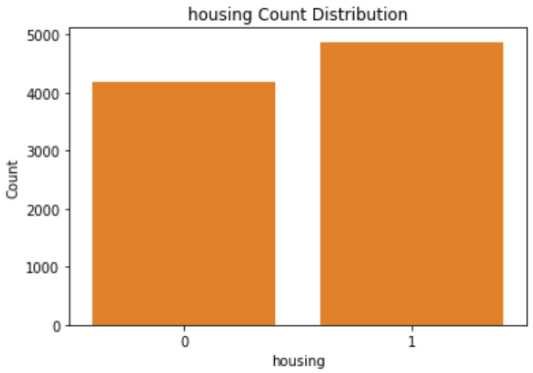
```
df.housing.replace(('yes', 'no'), (1, 0), inplace=True)
```

```
df.drop(df[(df['housing'] == "unknown")].index, inplace = True)
```

After:

```
df.drop(df[(df['housing'] == "unknown")].index, inplace = True)
```

```
uniform_color=sns.color_palette()[1]
sns.countplot(data=df,x='housing',color=uniform_color)
plt.xlabel('housing')
plt.ylabel('Count')
plt.title('housing Count Distribution');
```



9- Transform 'housing' column to numerical and drop unknown values since it's few

Loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

```
df['loan'].value_counts()
```

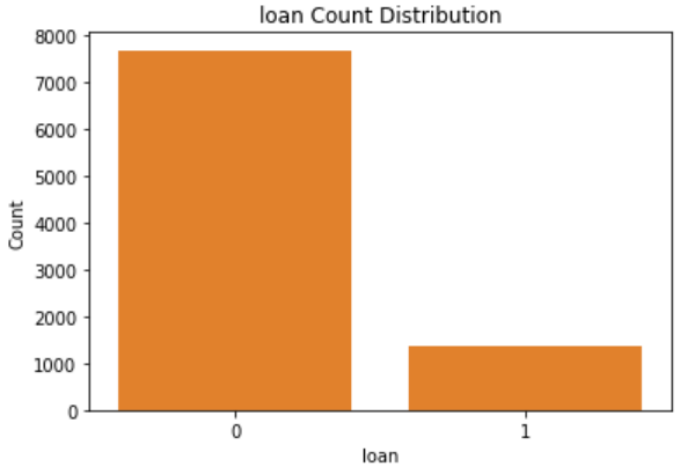
```
31]: no     7669
     yes    1378
     Name: loan, dtype: int64
```

```
df.loan.replace(('yes', 'no'), (1, 0), inplace=True)
```

```
df.drop(df[(df['loan'] == "unknown")].index, inplace = True)
```

After:

```
uniform_color=sns.color_palette()[1]
sns.countplot(data=df,x='loan',color=uniform_color)
plt.xlabel('loan')
plt.ylabel('Count')
plt.title('loan Count Distribution');
```

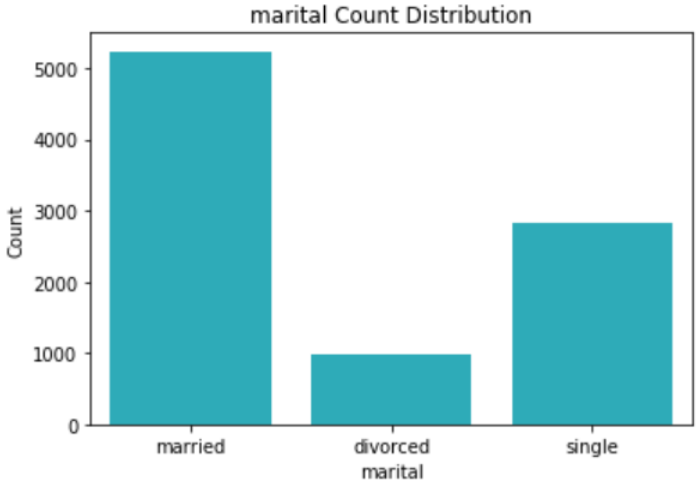


10- drop the unknown rows from ‘marital’ which represents the marital status, since they are few:

marital: marital status (categorical: 'divorced', 'married', 'single', 'unknown')

```
df.drop(df[(df['marital'] == "unknown")].index, inplace = True)
```

After:



11- Drop 'emp.var.rate','cons.price.idx','euribor3m','nr.employed','cons.conf.idx' columns

drop some of the columns that we won't use and sren't useful:

```
df=df.drop(['emp.var.rate','cons.price.idx','euribor3m','nr.employed','cons.conf.idx'],axis=1)
```

- **There are some questions that we try to answer to better understand the data:**

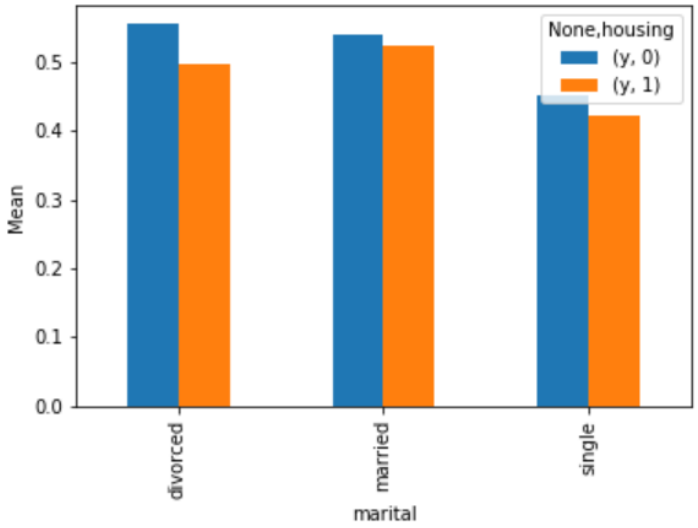
Q1 is the Marital with Housing associated with the if the client will subscribe a term deposit

```
df["y"]=pd.get_dummies(df["y"]);
df.groupby(['marital','housing'])[["y"]].mean().unstack().plot(kind='bar').set_ylabel('Mean');
df.groupby(['marital','housing'])[["y"]].mean()
#married > 1 , single > 2,divorced > 3
```

44]:

marital	housing	y
divorced	0	0.555102
	1	0.497992
married	0	0.539004
	1	0.523309
single	0	0.452305
	1	0.421530

The result:

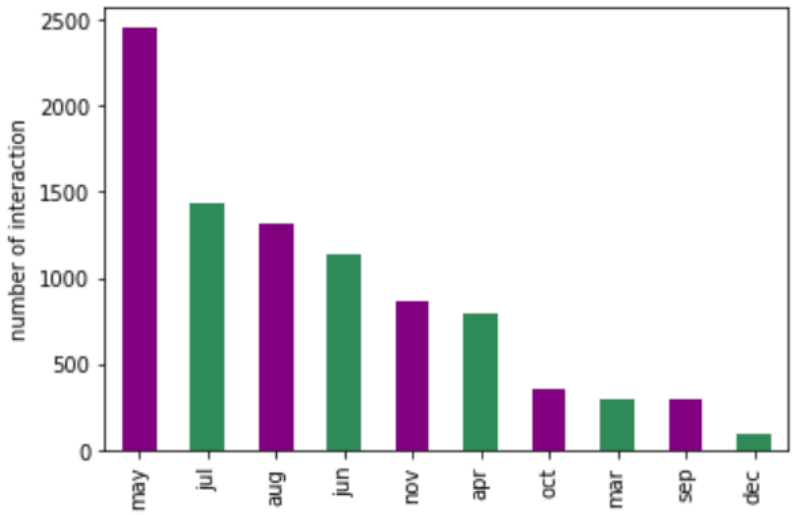


- The plot shows that marital wasn't really that affective on the housing nor 'y' column

Q2 the Most month that hse interaction on of the data

```
gender = df['month'].value_counts().plot.bar(color=('purple','seagreen'));
gender.set_xticklabels(["may", "jul","aug","jun","nov","apr","oct","mar","sep","dec"]);
gender.set_ylabel("number of interaction");
```

The result:

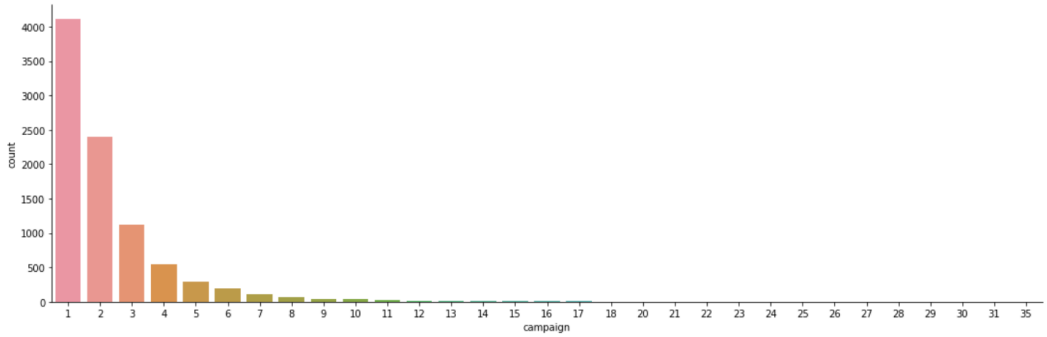


- Plot of ‘month’ column which represents the last contact month of year

- **Plots**:

1- Plot the ‘campaign’ column which represents the number of contacts performed during this campaign and for this client

```
sns.catplot('campaign',kind = 'count',data = df,aspect =3);
```



- The result shows that most of clients got called only one time for each campaign

2- The Unique values for each column:

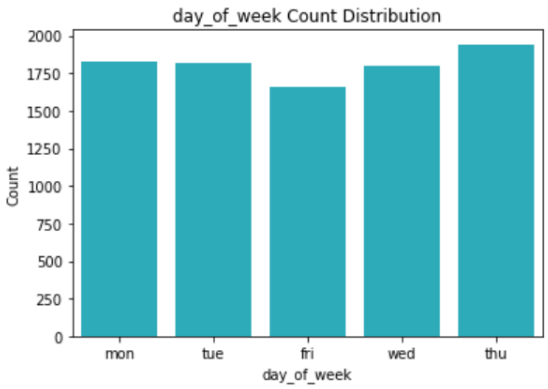
```
print("--------------")
print(df["job"].unique())
print("--------------")
print(df["marital"].unique())
print("--------------")
print(df["education"].unique())
print("--------------")
print(df["housing"].unique())
print("--------------")
print(df["loan"].unique())
print("--------------")
print(df["duration"].unique())
print("--------------")
print(df["campaign"].unique())
print("--------------")
print(df["previous"].unique())
print("--------------")
print(df["y"].unique())
print("--------------")
```

Result:

```
--------------
['management' 'admin.' 'self-employed' 'blue-collar' 'technician'
 'services' 'retired' 'student' 'entrepreneur' 'housemaid' 'unemployed'
 'unknown']
--------------
['married' 'divorced' 'single' 'unknown']
--------------
['high.school' 'university.degree' 'basic.9y' 'professional.course'
 'basic.6y' 'basic.4y' 'unknown' 'illiterate']
--------------
[0 1]
--------------
[0 1]
--------------
[  28  142  136 ...  759 1556 1868]
--------------
[ 8  5 10  2  1  3  4  7 16  6  9 11 18 15 33 25 20 32 12 17 22 13 24 29
 28 14 21 26 19 40 23]
--------------
[0 1 2 4 3 6 5]
--------------
[0 1]
--------------
```

3- Plot of 'day_of_week' column, which represents the last contact day of the week

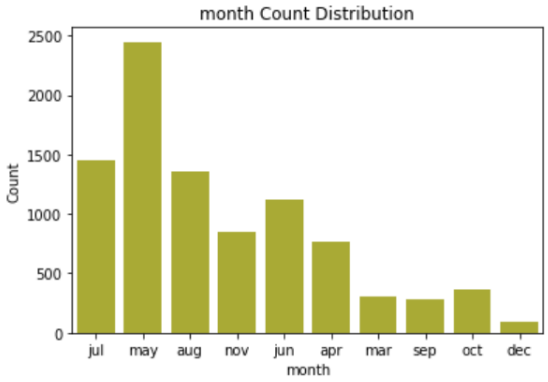
```
uniform_color=sns.color_palette()[9]
sns.countplot(data=df,x='day_of_week',color=uniform_color)
plt.xlabel('day_of_week')
plt.ylabel('Count')
plt.title('day_of_week Count Distribution');
```



- The day of the last contact of the week are similar, but Thursday is the highest

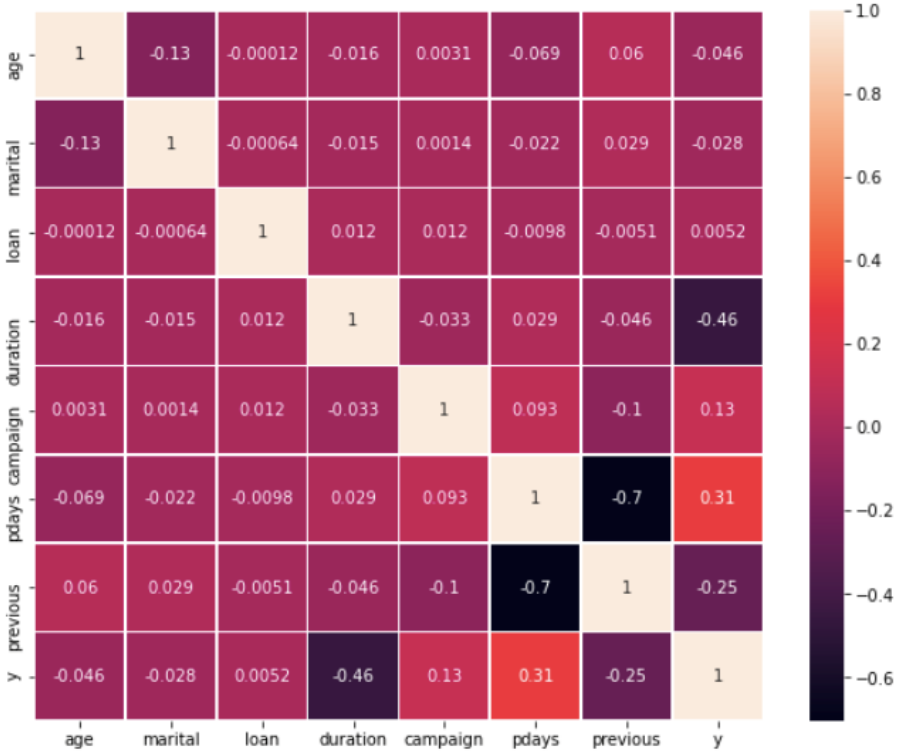
4- Plot of 'month' column which represents the last contact month of year

```
uniform_color=sns.color_palette()[8]
sns.countplot(data=df,x='month',color=uniform_color)
plt.xlabel('month')
plt.ylabel('Count')
plt.title('month Count Distribution');
```



- The highest month that has the last contact of the years is may

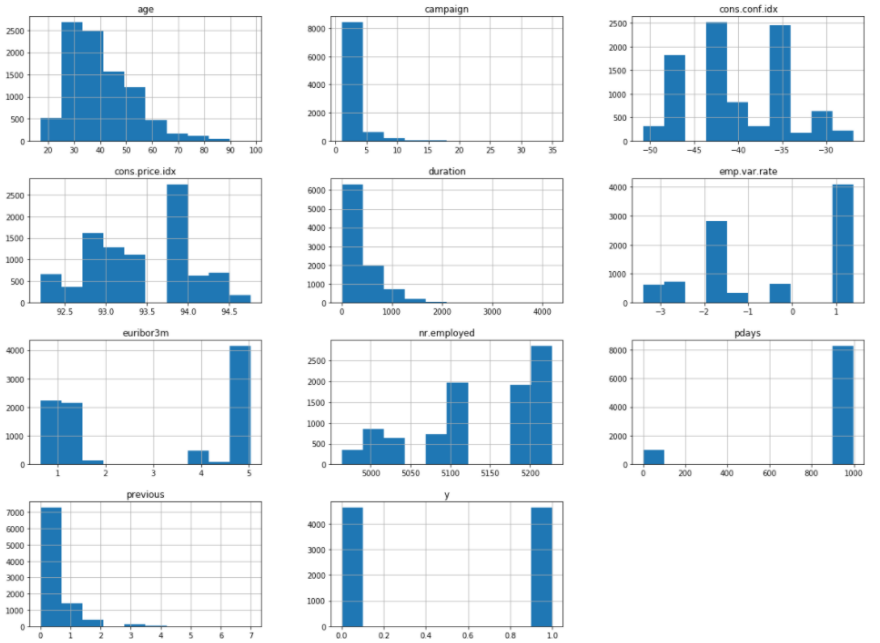
5- Heat map



- There is strong relation between pdays and y, also campaign and loan the rest columns has less connection

9- histogram

```
df.hist(figsize=(20,15));
```



- the histogram shows that on the age column the highest age is from 30-40
- The result shows that most of clients got called only one time for each campaign
- Most of the result shows that there are some biased on the data

III. Methodology

a) Explain the ML algorithm used

1- logistic regression which is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). logistic regression is a predictive analysis. it used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

2- SVM which is a supervised algorithm which can be used for both classification or regression challenges. Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and that what we are going for..

We choose these two models to train the data with and predict the result as two classes that will let us know which columns (variables) has the most effect on whether the client will subscribe a term deposit after a marketing campaigns of banking institution.

b) Implementation (with screenshots)

- **SVM**

First we split the columns and assigned the features and the target, then we split the processed dataset into a training set and a test set. The training set will be where we conduct cross validation (CV) for model selection, whereas the test set is unaccusable during model selection and serves as a final step to evaluate model performance.:

```
df_features=df[['age', 'loan', 'duration', 'campaign', 'pdays', 'previous']]
df_target=df[['y']].values
#df_features=df3.drop(columns=['y'],axis=1).values
x1_train, x1_test, y1_train, y1_test = train_test_split(df_features, df_target, test_size = 0.3, random_state = 0)
```

Before we start training, we will standardize the numeric variables:

```
sc = StandardScaler()
x1_train = sc.fit_transform(x1_train)
x1_test = sc.transform(x1_test)
```

Then we start training the data

```
from sklearn import svm
linear_svc = svm.SVC(kernel='linear',probability=True)#LinearSVC()
linear_svc.fit(x1_train, y1_train)
Y_pred = linear_svc.predict(x1_test)
acc_linear_svc = linear_svc.score(x1_train, y1_train)
print("Prediction Accuracy: ",acc_linear_svc)
print("the targer class " ,linear_svc.classes_)
print("the intercept ",linear_svc.intercept_)
print("the coefficients",linear_svc.coef_)
```

The result:

```
Prediction Accuracy:  0.7885985748218527
the targer class  [0 1]
the intercept  [-0.18983739]
the coefficients [[-0.09144328  0.03942813 -1.57723382  0.14856483  0.53273174 -0.21659487]]
```

Show the prediction of SVM:

```
#linear_svc = LinearSVC()
#linear_svc.fit(x1_train, y1_train)


Y_pred = linear_svc.predict(x1_test)
acc_linear_svc = linear_svc.score(x1_train, y1_train)
print("Prediction Accuracy: ",acc_linear_svc)
print("Accuracy:",metrics.accuracy_score(y1_test, Y_pred))
print("Precision:",metrics.precision_score(y1_test, Y_pred))
print("Recall:",metrics.recall_score(y1_test, Y_pred))
```

The result:

```
Prediction Accuracy:  0.7885985748218527
Accuracy: 0.7783524196527521
Precision: 0.7388493859082095
Recall: 0.8536221060492906
```

The Classification report:

```
print(classification_report(y1_test, Y_pred))
```

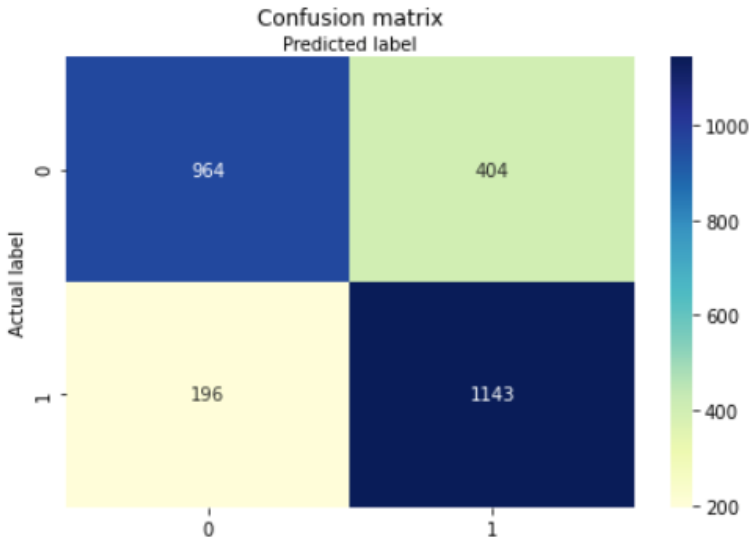
```
              precision    recall  f1-score   support

           0       0.83      0.70      0.76      1368
           1       0.74      0.85      0.79      1339

    accuracy                           0.78      2707
   macro avg       0.78      0.78      0.78      2707
weighted avg       0.79      0.78      0.78      2707
```

Confusion matrix:

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y1_test, Y_pred)
cnf_matrix
```

```
array([[ 964,  404],
       [ 196, 1143]])
```

confusion Matrix



Logistic Regression:

we start training the data then print the result:

```
logreg2=LogisticRegression(C=100,penalty="l2", max_iter=10000)
logreg2.fit(x1_train,y1_train)
print("score",logreg2.score(x1_test,y1_test))
print("the targer class " ,logreg2.classes_)
print("the intercept ",logreg2.intercept_)
print("the coefficients",logreg2.coef_)
```

```
score 0.7790912449205762
the targer class  [0 1]
the intercept  [-0.32061125]
the coefficients [[-0.14716768  0.05700535 -1.97275906  0.30085308  0.77708142 -0.34399254]]
```

The Classification report:

```
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import f1_score
from sklearn.metrics import average_precision_score

predictions = logreg2.predict(x1_test)

#print("Confusion Matrix:")
#print(confusion_matrix(y_test, predictions))

print("Classification Report")
print(classification_report(y1_test, predictions))
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.83      0.71      0.77      1368
           1       0.74      0.85      0.79      1339

    accuracy                           0.78      2707
   macro avg       0.78      0.78      0.78      2707
weighted avg       0.78      0.78      0.78      2707
```

Prediction of test sample:

```
print("Accuracy:",metrics.accuracy_score(y1_test, predictions))
print("Precision:",metrics.precision_score(y1_test, predictions))
print("Recall:",metrics.recall_score(y1_test, predictions))
```

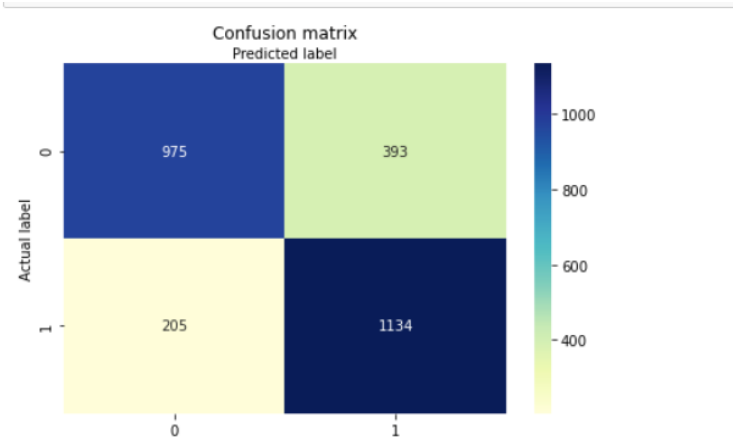
```
Accuracy: 0.7790912449205762
Precision: 0.7426326129666012
Recall: 0.8469006721433906
```

Confusion matrix:

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y1_test, predictions)
cnf_matrix
```

```
array([[ 975,  393],
       [ 205, 1134]])
```

Confusion matrix:



c) What are the initial parameters of the model if any?

- For SVM and Logistic regression model we choose the parameter that we think will affect the result and make better solution which were: 'age', 'loan', 'duration', 'campaign', 'previous', 'pdays'.

  also try different parameters and decided that we will choose the previous ones because it has the best accuracy

```
df_features=df[['age', 'loan', 'duration', 'campaign', 'previous' , 'pdays']] #
df_target=df[['y']].values
#df_features=df3.drop(columns=['y'],axis=1).values
x1_train, x1_test, y1_train, y1_test = train_test_split(df_features, df_target, test_size = 0.3, random_state = 0)
```

Intercept and coefficients for SVM:

```
Prediction Accuracy:  0.7885985748218527
the targer class  [0 1]
the intercept  [-0.18983739]
the coefficients [[-0.09144328  0.03942813 -1.57723382  0.14856483  0.53273174 -0.21659487]]
```

Intercept and coefficients for logistic regression:

```
score 0.7790912449205762
the targer class  [0 1]
the intercept  [-0.32061125]
the coefficients [[-0.14716768  0.05700535 -1.97275906  0.30085308  0.77708142 -0.34399254]]
```

IV. Evaluation and Results

a) Show your results (tables, figure, confusion matrices …etc)

Logistic Regression:

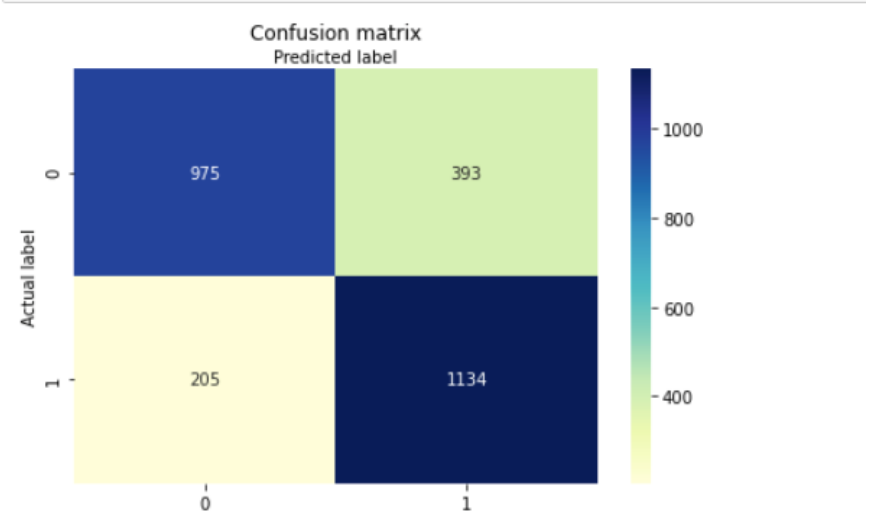
```
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import f1_score
from sklearn.metrics import average_precision_score

predictions = logreg2.predict(x1_test)

#print("Confusion Matrix:")
#print(confusion_matrix(y_test, predictions))

print("Classification Report")
print(classification_report(y1_test, predictions))
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.83      0.71      0.77      1368
           1       0.74      0.85      0.79      1339

    accuracy                           0.78      2707
   macro avg       0.78      0.78      0.78      2707
weighted avg       0.78      0.78      0.78      2707
```

```
print("Accuracy:",metrics.accuracy_score(y1_test, predictions))
print("Precision:",metrics.precision_score(y1_test, predictions))
print("Recall:",metrics.recall_score(y1_test, predictions))
```

```
Accuracy: 0.7790912449205762
Precision: 0.7426326129666012
Recall: 0.8469006721433906
```

- the accuracy is 0.779 which is pretty good result for the algorithm. Also the precision and recall were high too..



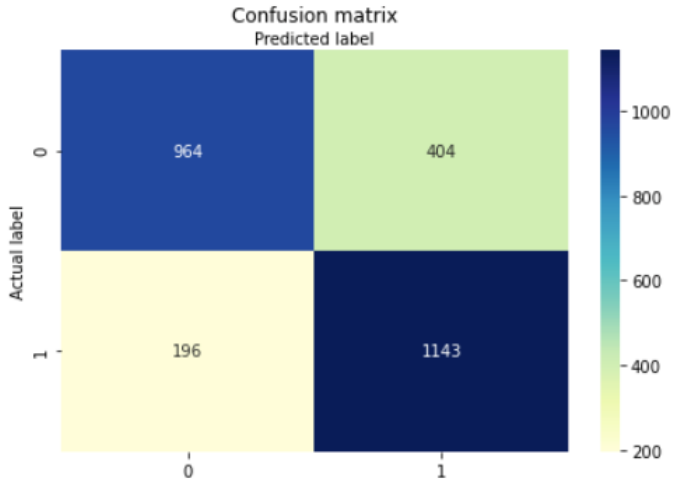
- Number of the class that were predicted as positive (=1) when the ‘y’ column was positive is 975 which is good result, also the number of class that were negative (=0) while the ‘y’ column is negative were 1134 which will predict high accuracy..

SVM

```
print(classification_report(y1_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.70      0.76      1368
           1       0.74      0.85      0.79      1339

    accuracy                           0.78      2707
   macro avg       0.78      0.78      0.78      2707
weighted avg       0.79      0.78      0.78      2707

Prediction Accuracy:  0.7885985748218527
the targer class  [0 1]
the intercept  [-0.18983739]
the coefficients [[-0.09144328  0.03942813 -1.57723382  0.14856483  0.53273174 -0.21659487]]
```

- the accuracy is 0.778 which is pretty good result for the algorithm but the logistic regression was better. Also the precision and recall were high too and close to the logistic regression result..



- Number of the class that were predicted as positive (=1) when the 'y' column was positive is 964 which is good result but less than the LR result, also the number of class that were negative (=0) while the 'y' column is negative were 1143 which is higher than the LR result but either way it will predict high accuracy..

b) Plot your output (visualization of error)

- **Logistic regression learning curve:**

```
import numpy as np
from sklearn.model_selection import learning_curve
sizes, training_scores, testing_scores = learning_curve(LogisticRegression(),
                                                        df_features, df_target, cv=10, scoring='neg_log_loss',
                                                        train_sizes=np.linspace(0.01, 0.8, 50))

mean_training = np.mean(-training_scores, axis=1)

# Mean and Standard Deviation of testing scores
mean_testing = np.mean(-testing_scores, axis=1)

# dotted blue line is for training scores and green line is for cross-validation score
plt.plot(sizes, mean_training, '--', color="b",  label="Training score")
plt.plot(sizes, mean_testing, color="g", label="Cross-validation score")

# Drawing plot
plt.title("LEARNING CURVE FOR Logistic Regression Classifier")
plt.xlabel("Training Set Size"), plt.ylabel("'MSE'"), plt.legend(loc="best")
plt.tight_layout()
plt.show()
```

- the learning curve shows that the Mean squared error is increase when we train it more, which means that there are more errors on the test data
- The underfitting is decreasing as we train the data more and as the increasing of the model complexity
- The mean squared error of Cross-validation score is decrease as we train the data and get less underfitting which is a good thing and will increase the accuracy of the logistic regression model

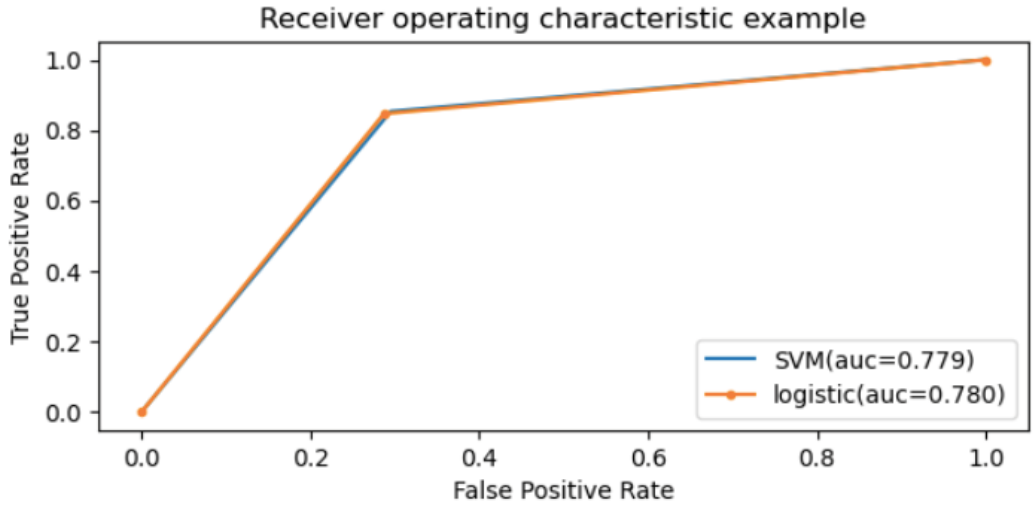
- **SVM learning curve**

```
import numpy as np
from sklearn.model_selection import learning_curve
sizes, training_scores, testing_scores = learning_curve(svm.SVC(kernel='linear',probability=True),
                                                        df_features, df_target, cv=10, scoring='neg_log_loss',
                                                        train_sizes=np.linspace(0.01, 0.8, 50))

mean_training = np.mean(-training_scores, axis=1)

# Mean and Standard Deviation of testing scores
mean_testing = np.mean(-testing_scores, axis=1)

# dotted blue line is for training scores and green line is for cross-validation score
plt.plot(sizes, mean_training, '--', color="b",  label="Training score")
plt.plot(sizes, mean_testing, color="g", label="Cross-validation score")

# Drawing plot
plt.title("LEARNING CURVE FOR Logistic Regression Classifier")
plt.xlabel("Training Set Size"), plt.ylabel("'MSE'"), plt.legend(loc="best")
plt.tight_layout()
plt.show()
```

c) Which evaluation techniques is better and why?

- According to the result we got for each method, they both has high accuracy and they got pretty close results, the accuracy of logistic regression was = 0.779 and for the SVM was = 0.778. so the logistic regression is better than the SVM
- There is ROC represents how the methods preform in different parts, and how their performance is pretty similar but the logistic regression is slightly better than the SVM

```
logustic_fpr,logustic_tpr,threshold = roc_curve(y1_test,predictions)
auc_logistic=auc(logustic_fpr,logustic_tpr)



svm_fpr,svm_tpr,threshold = roc_curve(y1_test,Y_pred)
auc_svm=auc(svm_fpr,svm_tpr)
plt.figure(figsize=(7,3),dpi=100)
plt.plot(svm_fpr,svm_tpr,linestyle='-',label='SVM(auc=%0.3f)'%auc_svm)
plt.plot(logustic_fpr,logustic_tpr,marker='.',label='logistic(auc=%0.3f)'%auc_logistic)

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend()
plt.show()
```



V. Analysis and conclusion

a) Explain your results in analytical way instead of numbers. Meaning, why do you think one algorithm performed that way and the other didn't?

- we got a pretty good accuracy for both methods, which means that the parameters: 'age', 'loan', 'duration', 'campaign', 'previous', 'pdays'. will affect the result and make a good solution which will help the banks with their future marketing campaigns.