Data Science Internship

Week 10: Data Science Project: Bank Marketing (Campaign)

Exploratory Data Analysis

Group Name: One
Name: Hira Fahim
Email: hirashahidd26@yahoo.com
Country: United Kingdom
Company: Unemployed
Specialization: Data Science
Batch Code: LISUM19
Submission Date: 13th April 2023
Submitted to: Data Glacier

Table of Contents

1. Problem Description

ABC Bank wants to sell its term deposit product to customers and before launching the product they want to develop a model which help them in understanding whether a particular customer will buy their product or not (based on customer's past interaction with bank or other Financial Institution).

2. Dataset Information

The data is related with direct marketing campaigns of a Portuguese 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.

3. Data understanding

- Shape of the dataset (Number of rows and columns)

```
In [18]: # no of rows and columns
         d.shape

Out[18]: (45211, 17)
```

Number of rows = 45211
Number of columns = 17

- Datatype of Columns and Non-null values

```
In [19]: # Datatypes of columns and non-null values
         d.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

- Numerical and categorical Features

```
Numeric Features:
Index(['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous'], dtype='object')
==============================================================================
Categorical Features:
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'poutcome', 'y'],
      dtype='object')
```

4. Exploratory Data Analysis

4.1. Drop Duplicate rows.

```
# Remove duplicate rows
d=d.drop_duplicates()
d
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

45211 rows × 17 columns

4.2. Drop unnecessary columns.

```
# 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 be discarded for a realistic predictive model.
d= d.drop(['duration'], axis=1)
```

```
d.shape
```

```
(45211, 16)
```

4.3. Change datatype of categorical columns

```
# change datatype of categorical columns into "category"
d["job"]=d["job"].astype("category")
d["marital"]=d["marital"].astype("category")
d["education"]=d["education"].astype("category")
d["default"]=d["default"].astype("category")
d["housing"]=d["housing"].astype("category")
d["loan"]=d["loan"].astype("category")
d["contact"]=d["contact"].astype("category")
d["month"]=d["month"].astype("category")
d["poutcome"]=d["poutcome"].astype("category")
d["y"]=d["y"].astype("category")
```

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45211 entries, 0 to 45210
Data columns (total 16 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  category
 2   marital    45211 non-null  category
 3   education  45211 non-null  category
 4   default    45211 non-null  category
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  category
 7   loan       45211 non-null  category
 8   contact    45211 non-null  category
 9   day        45211 non-null  int64
 10  month      45211 non-null  category
 11  campaign   45211 non-null  int64
 12  pdays      45211 non-null  int64
 13  previous   45211 non-null  int64
 14  poutcome   45211 non-null  category
 15  y          45211 non-null  category
dtypes: category(10), int64(6)
memory usage: 2.8 MB
```

5. Univariate Analysis

Descriptive analysis (or univariate analysis) provides an understanding of the characteristics of each attribute of the dataset. It also offers important evidence for feature selection in a later state.

5.1. Description of Data

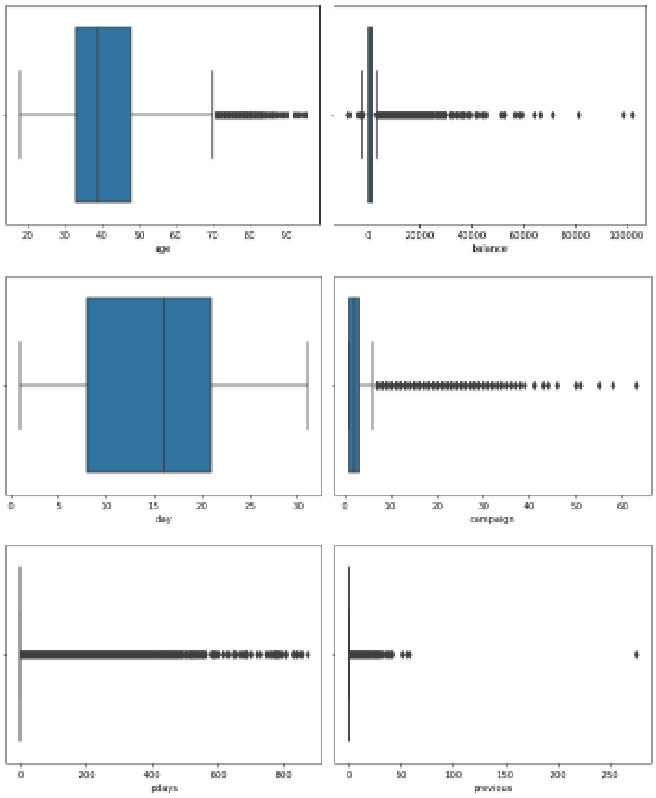
```
# Description of numerical columns
d.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

5.2. Boxplot for Numerical Attributes

```
# Function to plot boxplots
def plot_box_plots(dataframe):
    numeric_columns = numeric_features(dataframe)
    dataframe = dataframe[numeric_columns]

    for i in range(0,len(numeric_columns),2):
        if len(numeric_columns) > i+1:
            plt.figure(figsize=(10,4))
            plt.subplot(121)
            sns.boxplot(dataframe[numeric_columns[i]])
            plt.subplot(122)
            sns.boxplot(dataframe[numeric_columns[i+1]])
            plt.tight_layout()
            plt.show()

        else:
            sns.boxplot(dataframe[numeric_columns[i]])
```

```
plot_box_plots(d)
```



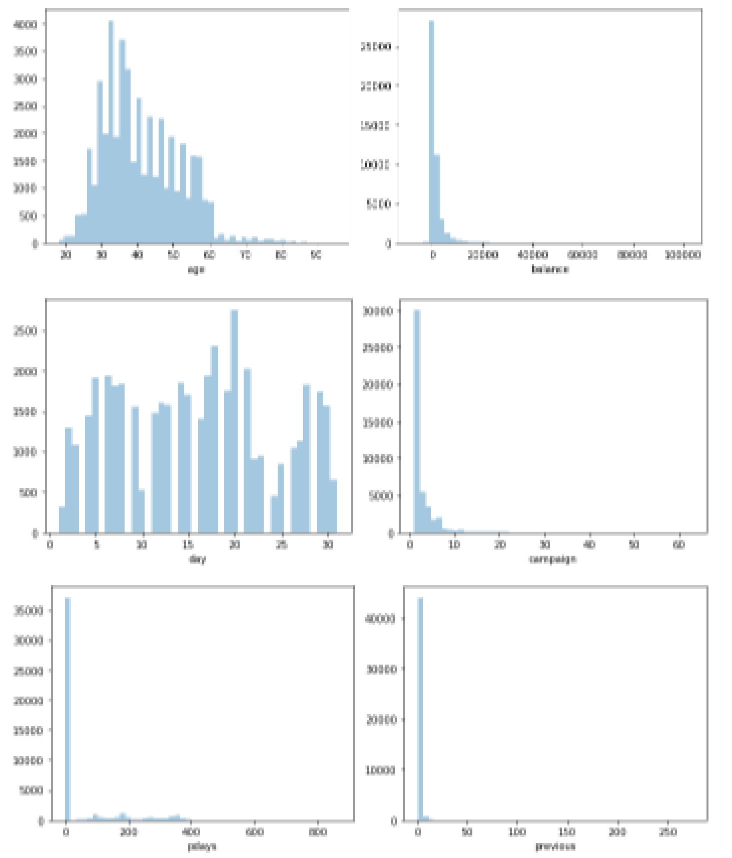
From **description** and **boxplot**, we can see there are outliers in numerical input variables like age, balance, campaign, pdays and previous. Pdays have most outliers comparatively.

5.3. Histogram for Numerical Attributes

For numerical attributes, generate the following statistical information and histograms. There are different distributions of values for different numerical attributes from the histograms, and some of the problematic issues begin appearing.

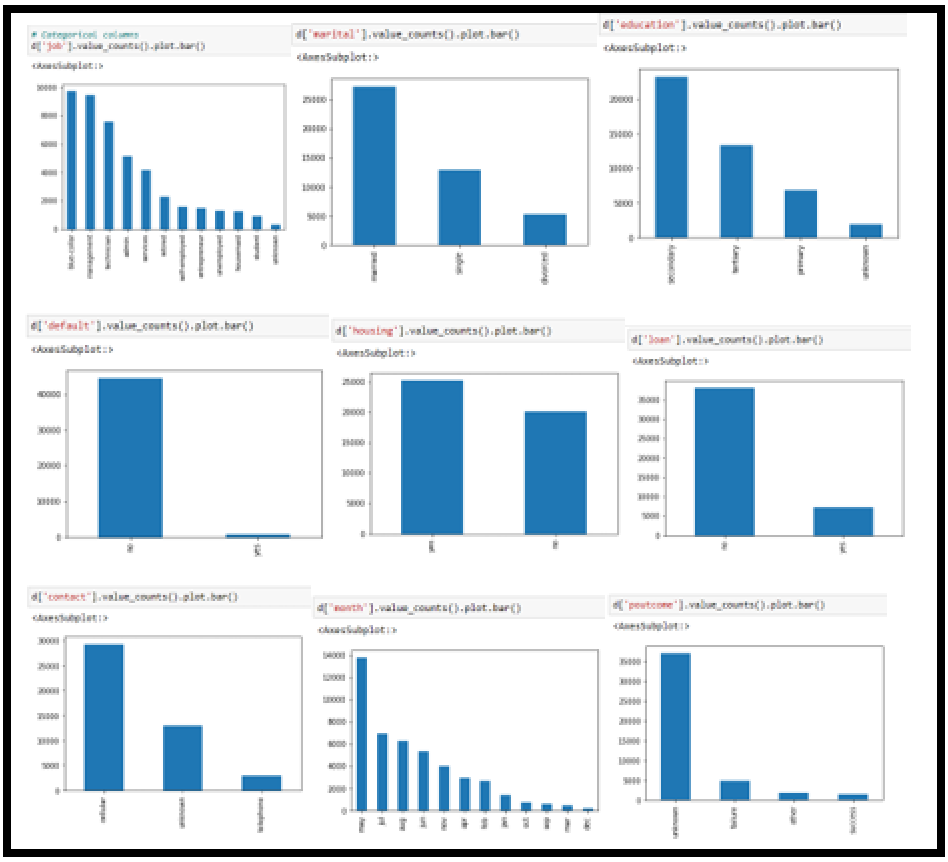
```
# Function to plot histograms
def plot_continuous_columns(dataframe):
    numeric_columns = numeric_features(dataframe)
    dataframe = dataframe[numeric_columns]

    for i in range(0,len(numeric_columns),2):
        if len(numeric_columns) > i+1:
            plt.figure(figsize=(10,4))
            plt.subplot(121)
            sns.distplot(dataframe[numeric_columns[i]], kde=False)
            plt.subplot(122)
            sns.distplot(dataframe[numeric_columns[i+1]], kde=False)
            plt.tight_layout()
            plt.show()

        else:
            sns.distplot(dataframe[numeric_columns[i]], kde=False)
```

```
plot_continuous_columns(d)
```



In Histogram, we can see input variables like age, balance, campaign, pdays and previous are **positively skewed**, and we can also see uneven distribution of data in day column.

5.4. Bar Plot for Categorical Attributes



In **Bar chart** of categorical columns, we see uneven distribution of data in all the input categorical columns.

6. Bivariate Analysis

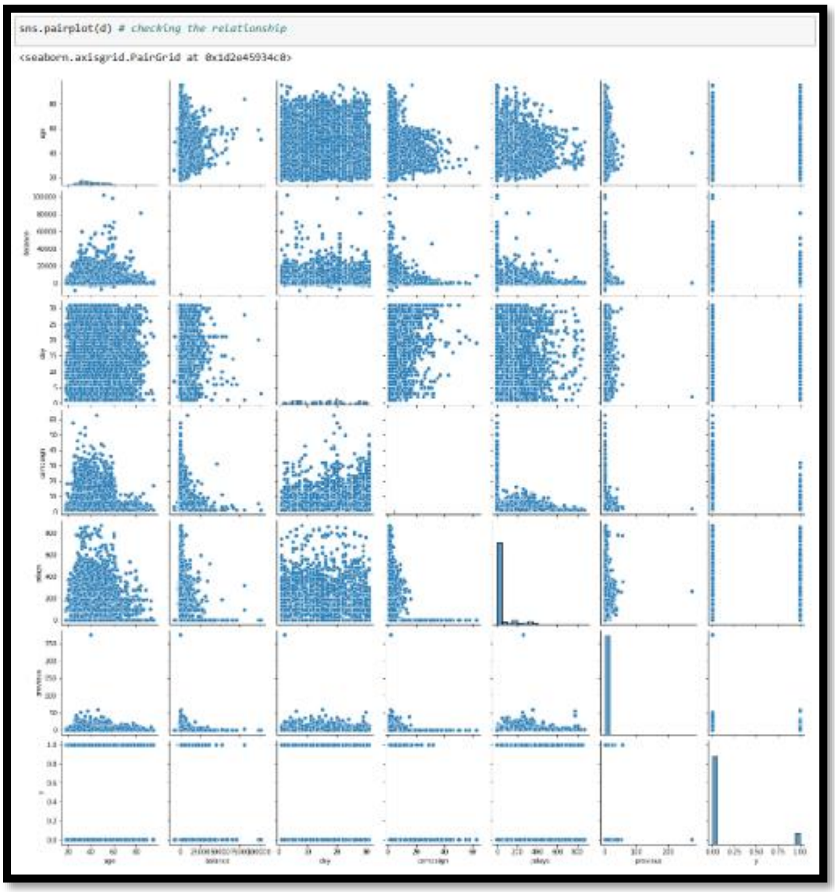
Correlation analysis (or bivariate analysis) examines the relationship between two attributes, say X and Y, and determines whether the two are correlated.

6.1. Correlation between Output and numerical input variables

```
d.corr()
```

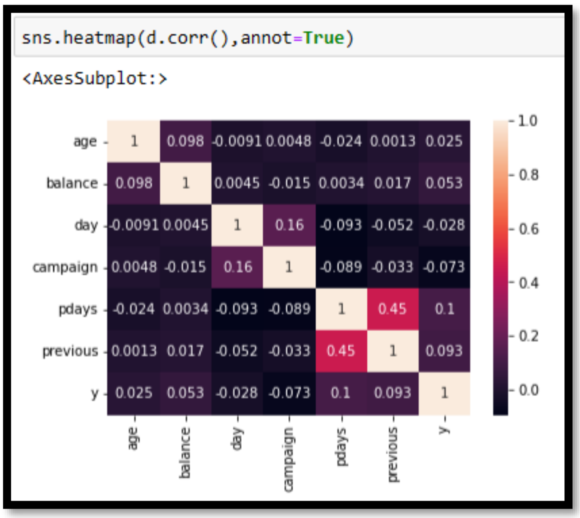
	age	balance	day	campaign	pdays	previous	y
age	1.000000	0.097783	-0.009120	0.004760	-0.023758	0.001288	0.025155
balance	0.097783	1.000000	0.004503	-0.014578	0.003435	0.016674	0.052838
day	-0.009120	0.004503	1.000000	0.162490	-0.093044	-0.051710	-0.028348
campaign	0.004760	-0.014578	0.162490	1.000000	-0.088628	-0.032855	-0.073172
pdays	-0.023758	0.003435	-0.093044	-0.088628	1.000000	0.454820	0.103621
previous	0.001288	0.016674	-0.051710	-0.032855	0.454820	1.000000	0.093236
y	0.025155	0.052838	-0.028348	-0.073172	0.103621	0.093236	1.000000

6.2. Pairplot



As per the **correlation coefficient** and **pairplot**, there is no strong correlation between numerical input variables and output variable.

6.3. Heatmap



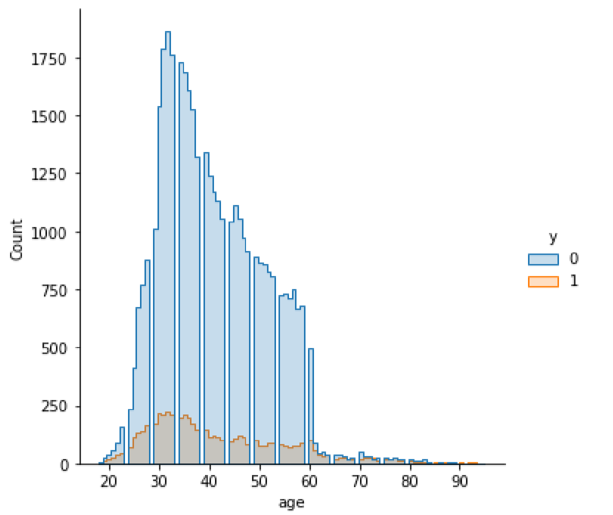
Here we see less correlation between numerical input variable and output variable.

7. Bivariate Analysis

7.1. Term deposit based on Age.

```
sns.displot(d, x="age", hue="y", element="step")
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f0f0cfd0>
```

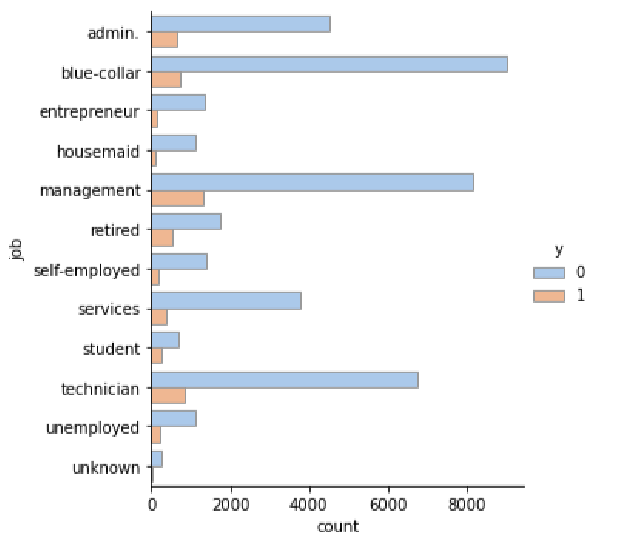


Here we see people between age 30-40 are more responsive towards term deposit.

7.2. Term deposit based on Job Type

```
sns.catplot(
    data=d, y="job", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f108e670>
```

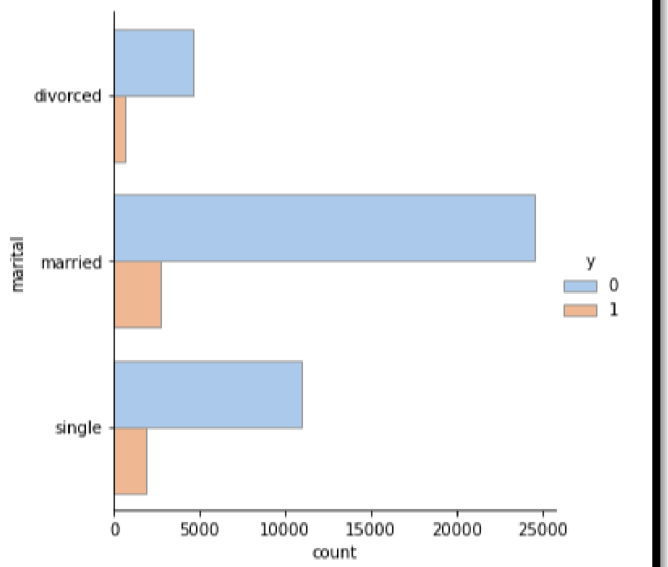


Here we see people with job related to 'management, blue-collar and technician' have subscribed for deposit.

7.3. Term deposit based on Marital Status

```
sns.catplot(
    data=d, y="marital", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f0edd9a0>
```

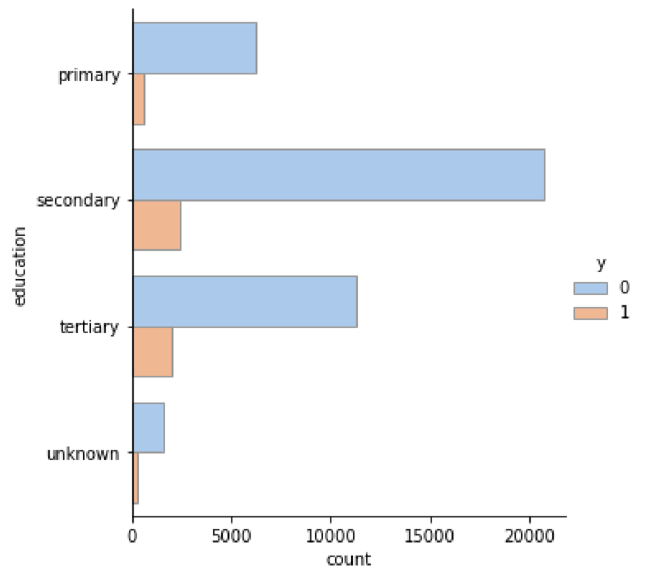


Married people are main contributor for deposit scheme.

7.4. Term deposit based on Education Level

```
sns.catplot(
    data=d, y="education", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f121d250>
```

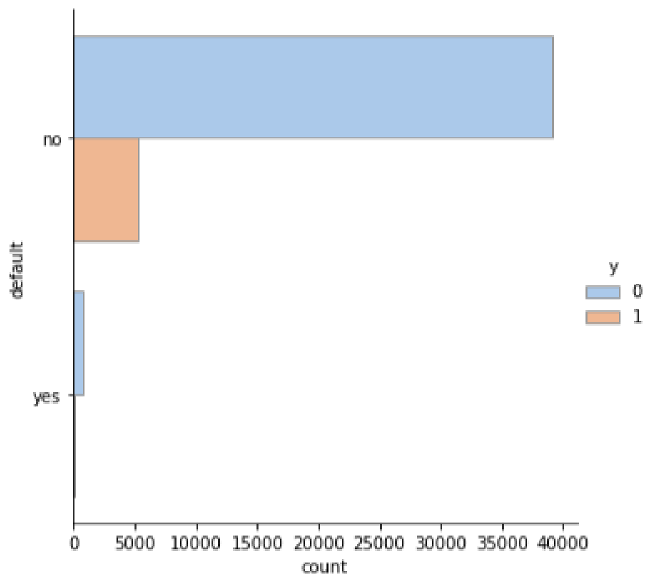


People with secondary and tertiary educational background are main contributors.

7.5. Term deposit based on Credit Default

```
sns.catplot(
    data=d, y="default", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f1087ee0>
```

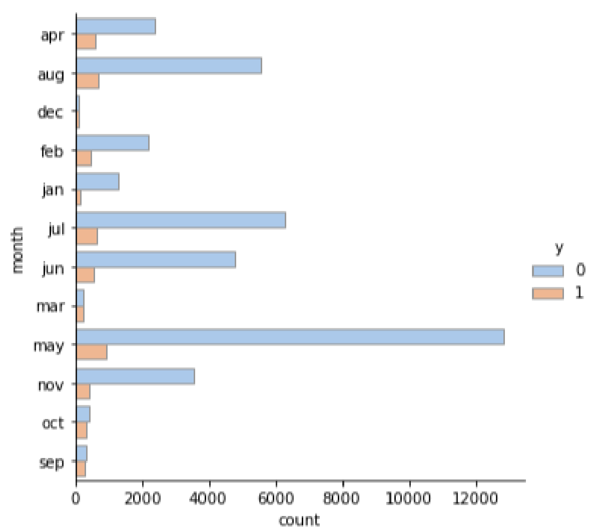


People with good credit history are more interested in term deposit.

7.6. Term deposit based on Month.

```
sns.catplot(
    data=d, y="month", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1c2e3214280>
```

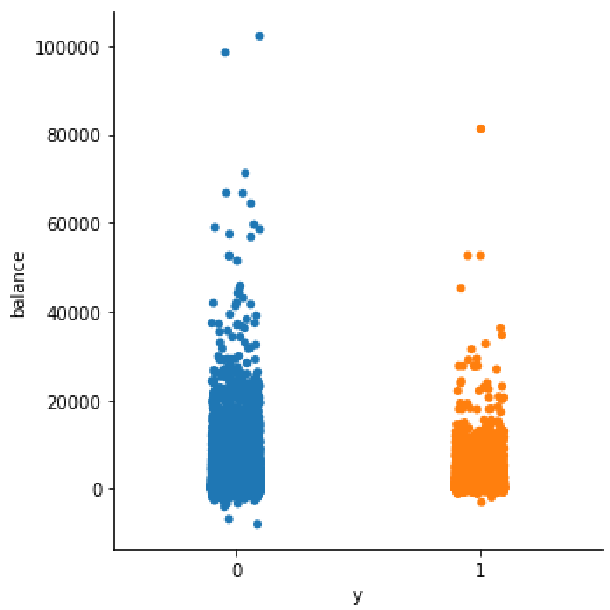


Investment in term deposit is fluctuating throughout the year but in the month of 'May' we have highest success rate.

7.7. Term deposit based on Balance.

```
sns.catplot(data=d, x="y", y="balance")
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f1a36af0>
```

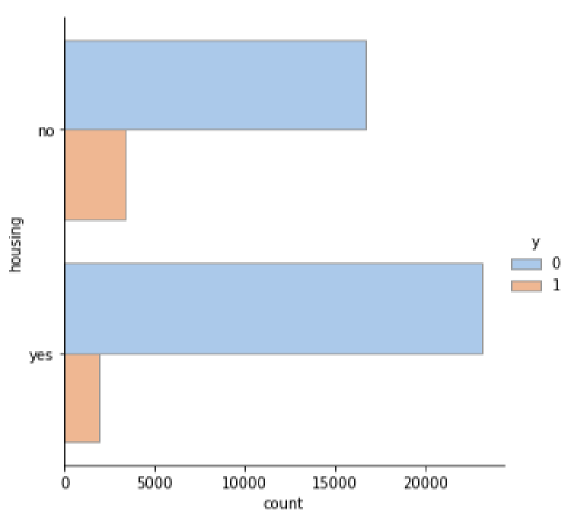


People with balance up to 20,000 are more interested in term deposit.

7.8. Term deposit based on Housing Loan

```
sns.catplot(
    data=d, y="housing", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f1c55f70>
```

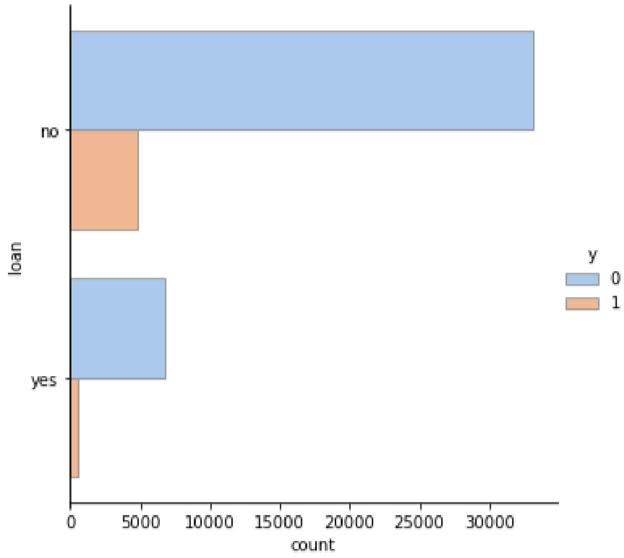


People with no housing scheme have subscribed for the term deposit.

7.9. Term deposit based on Personal Loan

```
sns.catplot(
    data=d, y="loan", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d2f1a3cf70>
```

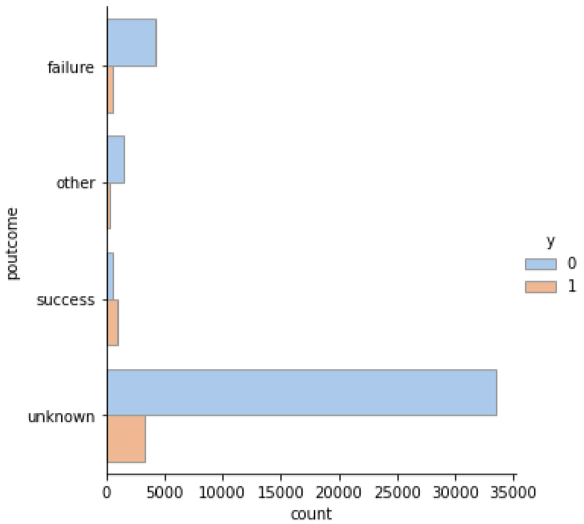


People with no personal loan are more interested in term deposit.

7.10. Term deposit based on outcome of Previous Campaign

```
sns.catplot(
    data=d, y="poutcome", hue="y", kind="count",
    palette="pastel", edgecolor=".6",)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d9864430a0>
```



From the Outcome of previous Campaign, if the outcome is Failure, then there is a less chance that client will subscribe to the term deposit. whereas if the outcome of previous Campaign is Success, then it is more likely that Client will subscribe to the term deposit.

8. Final Recommendations

- Most of the clients in the bank are contacted in the months of May, Jun, Jul and in Aug last year. Out of that, most of the clients contacted in the month of May and this is the month where clients are not interested to subscribe the term deposits. Very few of the clients are contacted in the months of march, sept and in Dec. It is better to Contact the clients more in these months.
- To increase the likelihood of subscription, the bank should re-evaluate the content and design of its current campaign, making it more appealing to its target customers. If the campaign is successful, then clients are more likely to subscribe for the term deposit.
- The bank could provide better banking services. For example, marital status and occupation reveal a customer's life stage while loan status indicates his/her overall risk profile. With this information, the bank can estimate when a customer might need to make an investment. In this way, the bank can better satisfy its customer demand by providing banking services for the right customer at the right time.
- The bank should target the right clients like clients with no personal and housing loan are much more interested in term deposit.