BIA 5302 – Machine Learning

Group Members:

- Daniel Arantes Ventura - N01468881
- Rutuja Bhagatsing Kadam - N01468983
- Mariana Reyes - N01468759
- Maria Paula Rodriguez - N01469279
- Mugdha Vivek Bhatwadekar- N01468894

Assignment 2

In [1]:

```
# packages and versions

from platform import python_version
print('Python and packages versions used in this Jupyter Notebook: ')
print('python    :', python_version())

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%reload_ext watermark
%watermark --iversions
```

```
Python and packages versions used in this Jupyter Notebook:
python    : 3.9.12
pandas    : 1.2.2
matplotlib: 3.5.1
numpy     : 1.20.1
seaborn   : 0.11.2
```

In [2]:

```
bank_df = pd.read_csv("bank-full.csv", sep = ";")
```

In [3]:

```
print('Number of instances: ', bank_df.shape[0])
print('Number of variables: ', bank_df.shape[1])
```

```
Number of instances:  45211
Number of variables:  17
```

In [4]:
```
bank_df.head()
```

Out[4]:

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

In [5]:
```
bank_df.columns
```

Out[5]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housin
g',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [6]:
```
bank_df.dtypes
```

Out[6]:
```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

Additional information from the dataset:

Bank client data:

1 - age (numeric) 2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar","self-employed","retired","technician","services") 3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed) 4 - education

(categorical: "unknown","secondary","primary","tertiary") 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: "unknown","telephone","cellular") 10 - day: last contact day of the month (numeric) 11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec") 12 - duration: last contact duration, in seconds (numeric)

Other attributes:

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact) 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: "unknown","other","failure","success")

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

Splitting into Categorical and Numerical Variables

In [7]:
```
num = ['age','balance','day','duration','campaign', 'pdays', 'previous']
```

In [8]:
```
cat = ['job','marital','education', 'default','housing','loan','contact','mor
```

In [9]:
```
bank_num = bank_df[num]
bank_num.sample(5)
```

Out[9]:

	age	balance	day	duration	campaign	pdays	previous
14325	34	344	14	56	3	-1	0
26706	41	666	20	106	3	183	3
24563	40	0	17	113	1	-1	0
16612	51	-238	24	63	1	-1	0
26431	31	328	20	89	1	-1	0

In [10]:
```
bank_cat = bank_df[cat]
bank_cat.sample(5)
```

Out[10]:

	job	marital	education	default	housing	loan	contact	month	poutcome
18472	unemployed	single	secondary	no	no	no	cellular	jul	unknown
9267	blue-collar	married	secondary	no	no	no	unknown	jun	unknown
15580	technician	single	secondary	no	no	yes	cellular	jul	unknown
6255	technician	divorced	secondary	no	yes	yes	unknown	may	unknown
44886	management	married	tertiary	no	no	no	cellular	sep	failure

Basic Statistics

In [11]:
```
bank_num.describe().round(2)
```

Out[11]:

	age	balance	day	duration	campaign	pdays	previous
count	45211.00	45211.00	45211.00	45211.00	45211.00	45211.00	45211.00
mean	40.94	1362.27	15.81	258.16	2.76	40.20	0.58
std	10.62	3044.77	8.32	257.53	3.10	100.13	2.30
min	18.00	-8019.00	1.00	0.00	1.00	-1.00	0.00
25%	33.00	72.00	8.00	103.00	1.00	-1.00	0.00
50%	39.00	448.00	16.00	180.00	2.00	-1.00	0.00
75%	48.00	1428.00	21.00	319.00	3.00	-1.00	0.00
max	95.00	102127.00	31.00	4918.00	63.00	871.00	275.00

In [12]:
```
#computing basic statistics plus the coefficient of variation
pd.DataFrame({'mean': bank_num.mean(),
'std dev': bank_num.std(),
'min': bank_num.min(),
'max': bank_num.max(),
'median': bank_num.median(),
'var coeficient':bank_num.std()/bank_num.mean()
}).round(2)
```

Out[12]:

	mean	std dev	min	max	median	var coeficient
age	40.94	10.62	18	95	39.0	0.26
balance	1362.27	3044.77	-8019	102127	448.0	2.24
day	15.81	8.32	1	31	16.0	0.53
duration	258.16	257.53	0	4918	180.0	1.00
campaign	2.76	3.10	1	63	2.0	1.12
pdays	40.20	100.13	-1	871	-1.0	2.49
previous	0.58	2.30	0	275	0.0	3.97

Histogram of Quantitative Variables

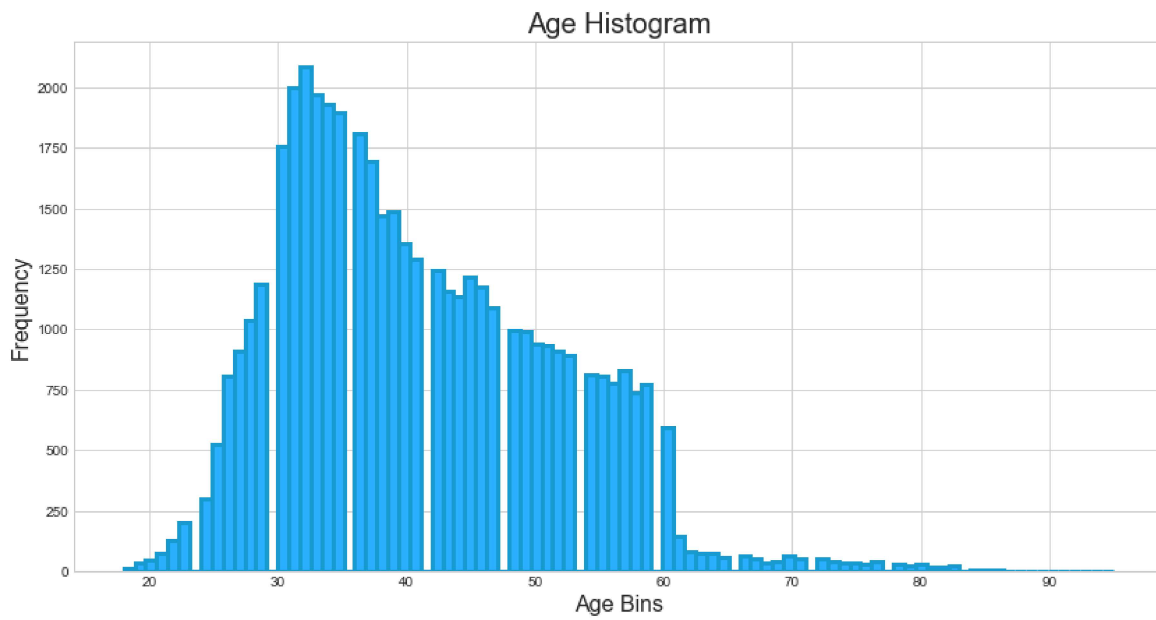
In [13]:
```
## Calculate Skewness
##· If the skewness is between -0.5 and 0.5, the data are fairly symmetrical.
##If the skewness is between -1 and — 0.5 or between 0.5 and 1, the data are
##If the skewness is less than -1 or greater than 1, the data are highly skew

bank_df.skew(axis = 0, skipna = True)
```

Out[13]:
```
age          0.684818
balance      8.360308
day          0.093079
duration     3.144318
campaign     4.898650
pdays        2.615715
previous    41.846454
dtype: float64
```

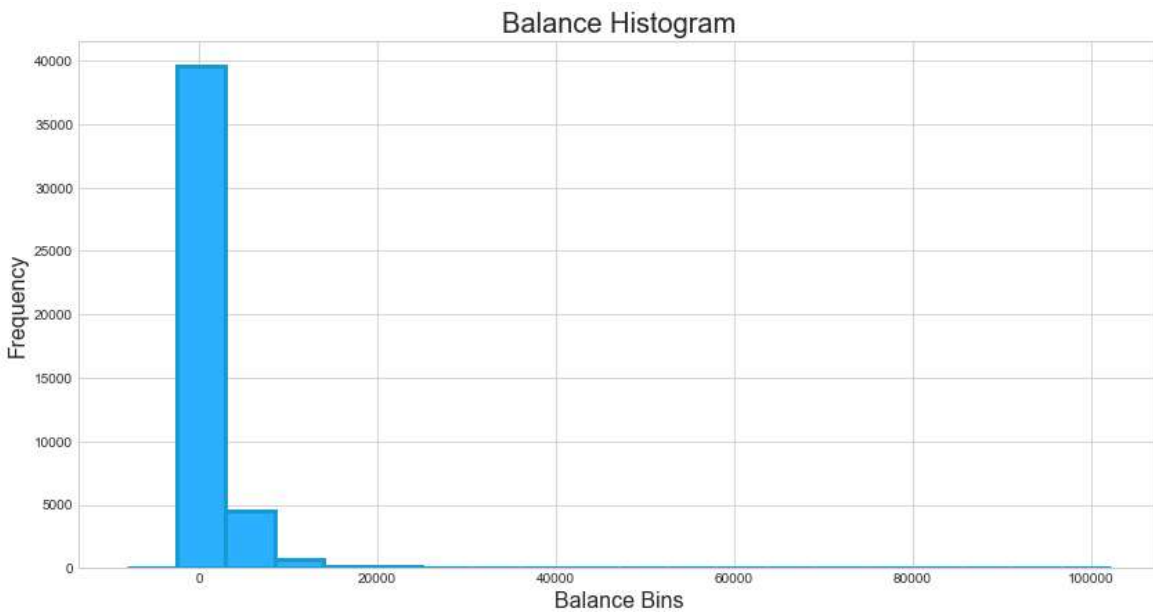
In [14]:

```
# "age" variable Histogram using Matplotlib
plt.figure(figsize=(14,7)) # Make it 14x7 inch
plt.style.use('seaborn-whitegrid') # nice and clean grid
plt.hist(bank_df.age,bins=90,facecolor = '#2ab0ff', edgecolor='#169acf', line
plt.title('Age Histogram',fontsize=20)
plt.xlabel('Age Bins',fontsize=16)
plt.ylabel('Frequency',fontsize=16)
plt.show()
```

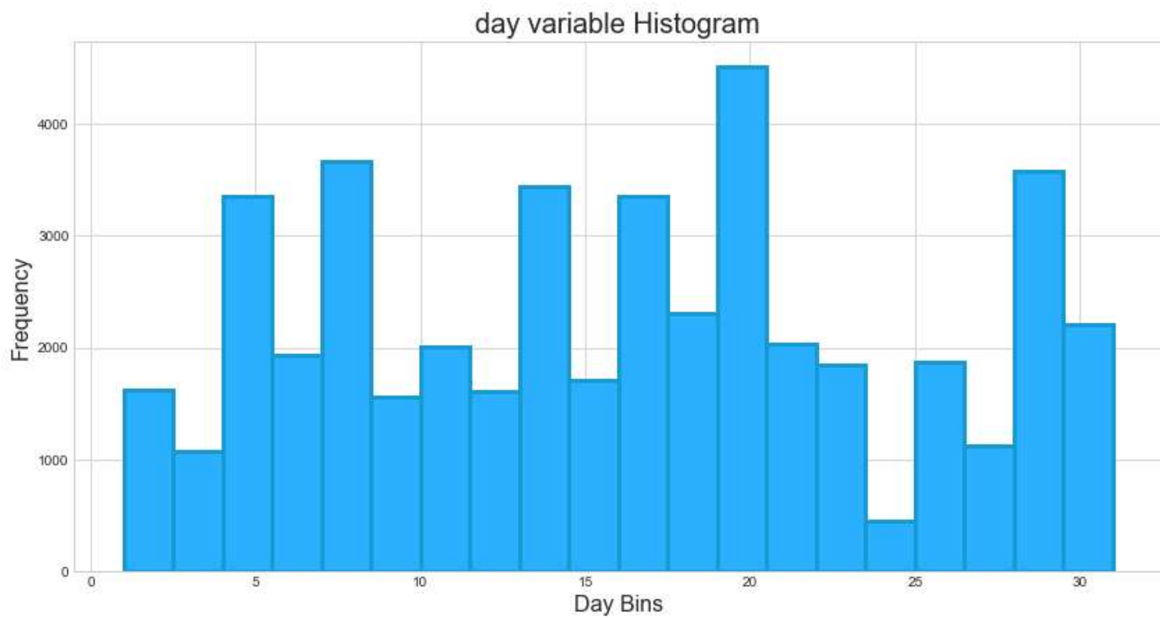


In [15]:

```
# "balance" variable Histogram using Matplotlib
plt.figure(figsize=(14,7)) # Make it 14x7 inch
plt.style.use('seaborn-whitegrid') # nice and clean grid
plt.hist(bank_df.balance,bins=20,facecolor = '#2ab0ff', edgecolor='#169acf',
plt.title('Balance Histogram',fontsize=20)
plt.xlabel('Balance Bins',fontsize=16)
plt.ylabel('Frequency',fontsize=16)
plt.show()
```

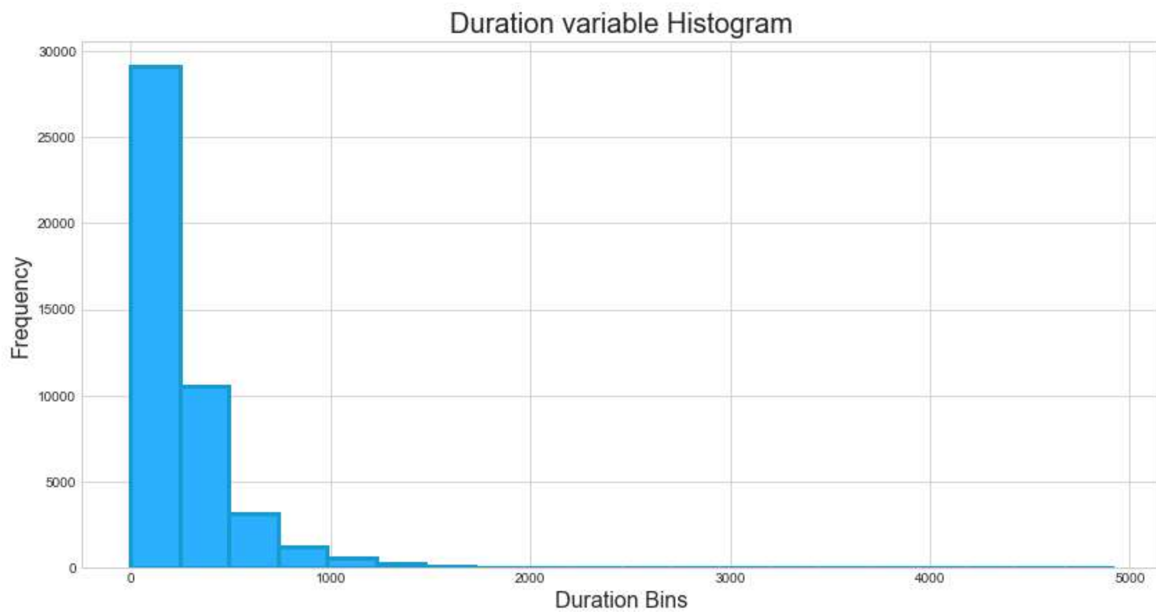


In [16]:
```
# "day" variable Histogram using Matplotlib
plt.figure(figsize=(14,7)) # Make it 14x7 inch
plt.style.use('seaborn-whitegrid') # nice and clean grid
plt.hist(bank_df.day,bins=20,facecolor = '#2ab0ff', edgecolor='#169acf', line
plt.title('day variable Histogram',fontsize=20)
plt.xlabel('Day Bins',fontsize=16)
plt.ylabel('Frequency',fontsize=16)
plt.show()
```



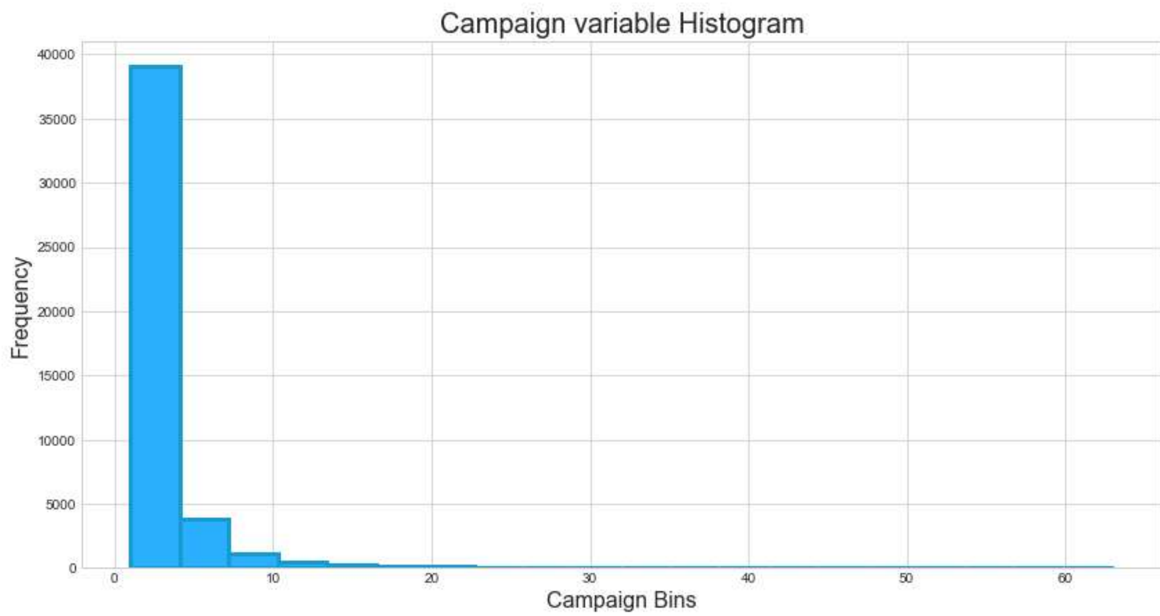
In [17]:

```
# "duration" variable Histogram using Matplotlib
plt.figure(figsize=(14,7)) # Make it 14x7 inch
plt.style.use('seaborn-whitegrid') # nice and clean grid
plt.hist(bank_df.duration,bins=20,facecolor = '#2ab0ff', edgecolor='#169acf',
plt.title('Duration variable Histogram',fontsize=20)
plt.xlabel('Duration Bins',fontsize=16)
plt.ylabel('Frequency',fontsize=16)
plt.show()
```



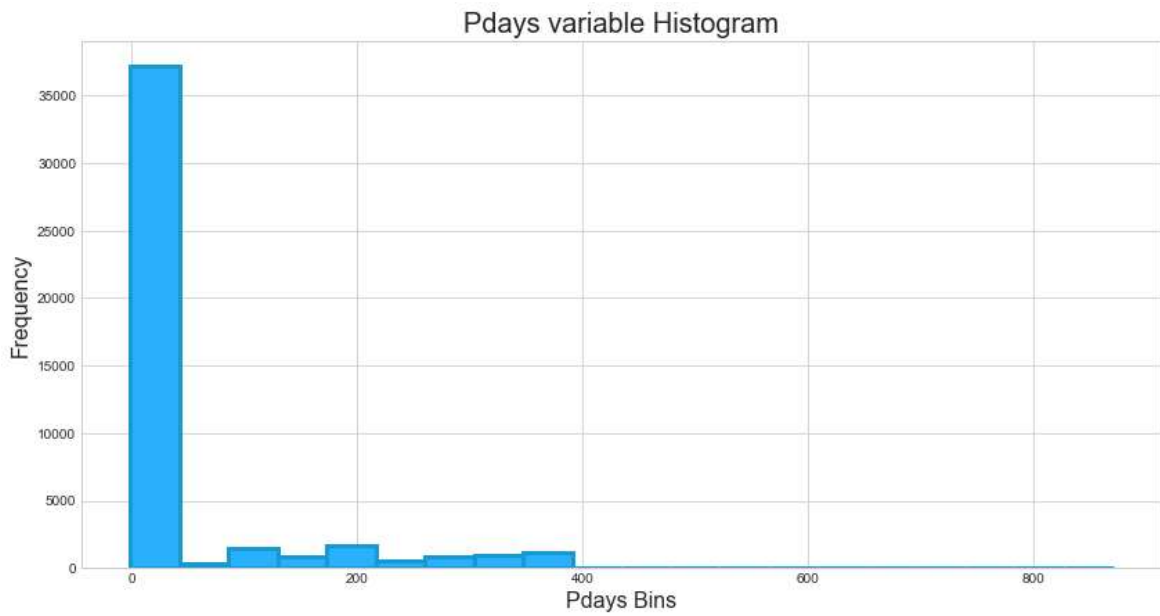
In [18]:

```
# "campaign" variable Histogram using Matplotlib
plt.figure(figsize=(14,7)) # Make it 14x7 inch
plt.style.use('seaborn-whitegrid') # nice and clean grid
plt.hist(bank_df.campaign,bins=20,facecolor = '#2ab0ff', edgecolor='#169acf',
plt.title('Campaign variable Histogram',fontsize=20)
plt.xlabel('Campaign Bins',fontsize=16)
plt.ylabel('Frequency',fontsize=16)
plt.show()
```



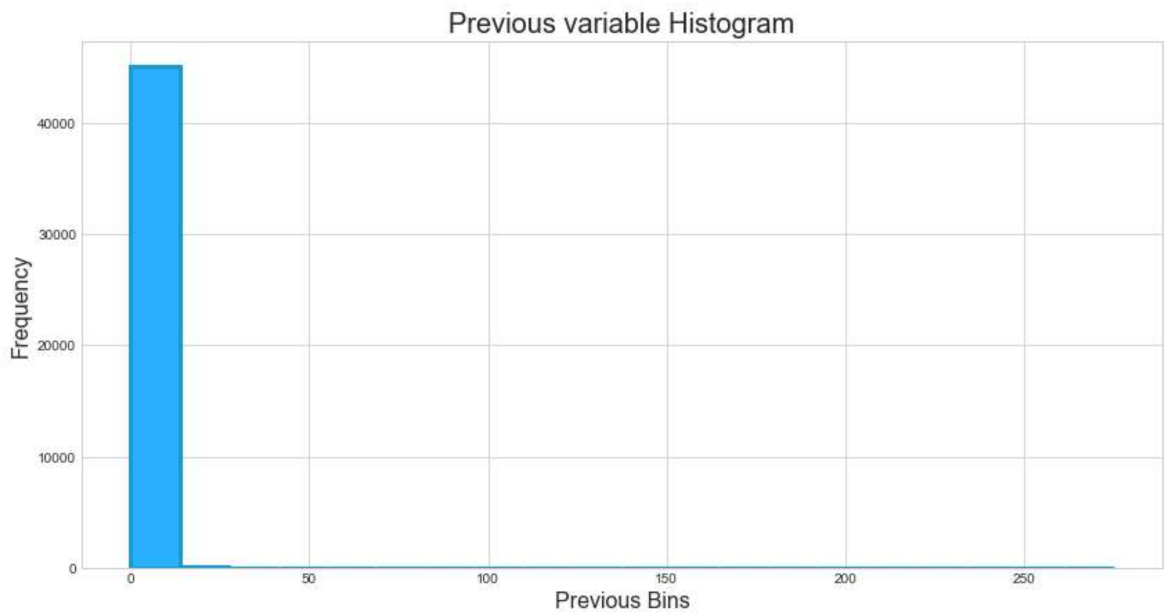
In [19]:

```
# "pdays" variable Histogram using Matplotlib
plt.figure(figsize=(14,7)) # Make it 14x7 inch
plt.style.use('seaborn-whitegrid') # nice and clean grid
plt.hist(bank_df.pdays,bins=20,facecolor = '#2ab0ff', edgecolor='#169acf', li
plt.title('Pdays variable Histogram',fontsize=20)
plt.xlabel('Pdays Bins',fontsize=16)
plt.ylabel('Frequency',fontsize=16)
plt.show()
```



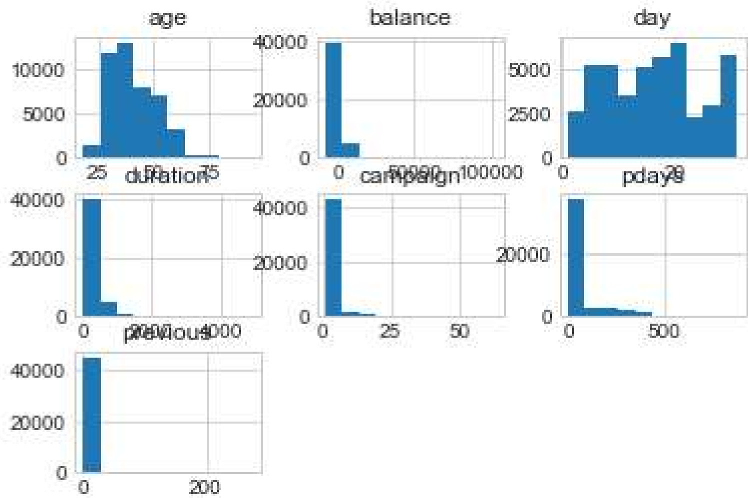
In [20]:

```
# "previous" variable Histogram using Matplotlib
plt.figure(figsize=(14,7)) # Make it 14x7 inch
plt.style.use('seaborn-whitegrid') # nice and clean grid
plt.hist(bank_df.previous,bins=20,facecolor = '#2ab0ff', edgecolor='#169acf',
plt.title('Previous variable Histogram',fontsize=20)
plt.xlabel('Previous Bins',fontsize=16)
plt.ylabel('Frequency',fontsize=16)
plt.show()
```



In [21]:
```
bank_df.hist()
```

Out[21]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'balance'}>,
        <AxesSubplot:title={'center':'day'}>],
       [<AxesSubplot:title={'center':'duration'}>,
        <AxesSubplot:title={'center':'campaign'}>,
        <AxesSubplot:title={'center':'pdays'}>],
       [<AxesSubplot:title={'center':'previous'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



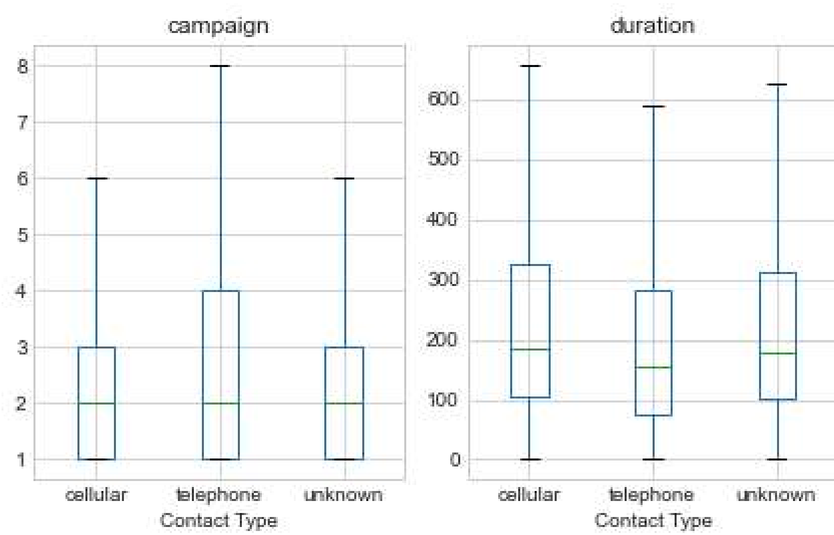
In [22]:
```
## Calculate Skewness
##· If the skewness is between -0.5 and 0.5, the data are fairly symmetrical.
##If the skewness is between -1 and – 0.5 or between 0.5 and 1, the data are
##If the skewness is less than -1 or greater than 1, the data are highly skew

bank_df.skew(axis = 0, skipna = True)
```

Out[22]:
```
age          0.684818
balance      8.360308
day          0.093079
duration     3.144318
campaign     4.898650
pdays        2.615715
previous    41.846454
dtype: float64
```

Boxplot Comparison of Two Variables

In [23]:
```
fig, axes = plt.subplots(nrows = 1, ncols = 2)
bank_df.boxplot(column='campaign', by='contact', ax=axes[0],showfliers=False)
bank_df.boxplot(column='duration', by='contact', ax=axes[1],showfliers=False)
for ax in axes:
    ax.set_xlabel('Contact Type')
    plt.suptitle('') # Suppress the overall title
    plt.tight_layout() #Increase the separation between the plot
```



Correlation

In [24]:
```
# list of boolean variables
bol = ['default', 'housing', 'loan', 'y']

# convert "yes" to 1 and "no" to 0.
dic = {"yes":1, "no":0}
for i in bol:
    bank_df[i] = bank_df[i].map(dic)
```

In [25]:
```
bank_df.head()
```

Out[25]:

	age	job	marital	education	default	balance	housing	loan	contact	day	mont
0	58	management	married	tertiary	0	2143	1	0	unknown	5	ma
1	44	technician	single	secondary	0	29	1	0	unknown	5	ma
2	33	entrepreneur	married	secondary	0	2	1	1	unknown	5	ma
3	47	blue-collar	married	unknown	0	1506	1	0	unknown	5	ma
4	33	unknown	single	unknown	0	1	0	0	unknown	5	ma

In [26]:
```
df = bank_df[['age','default','balance','housing','loan','day','duration','ca
```

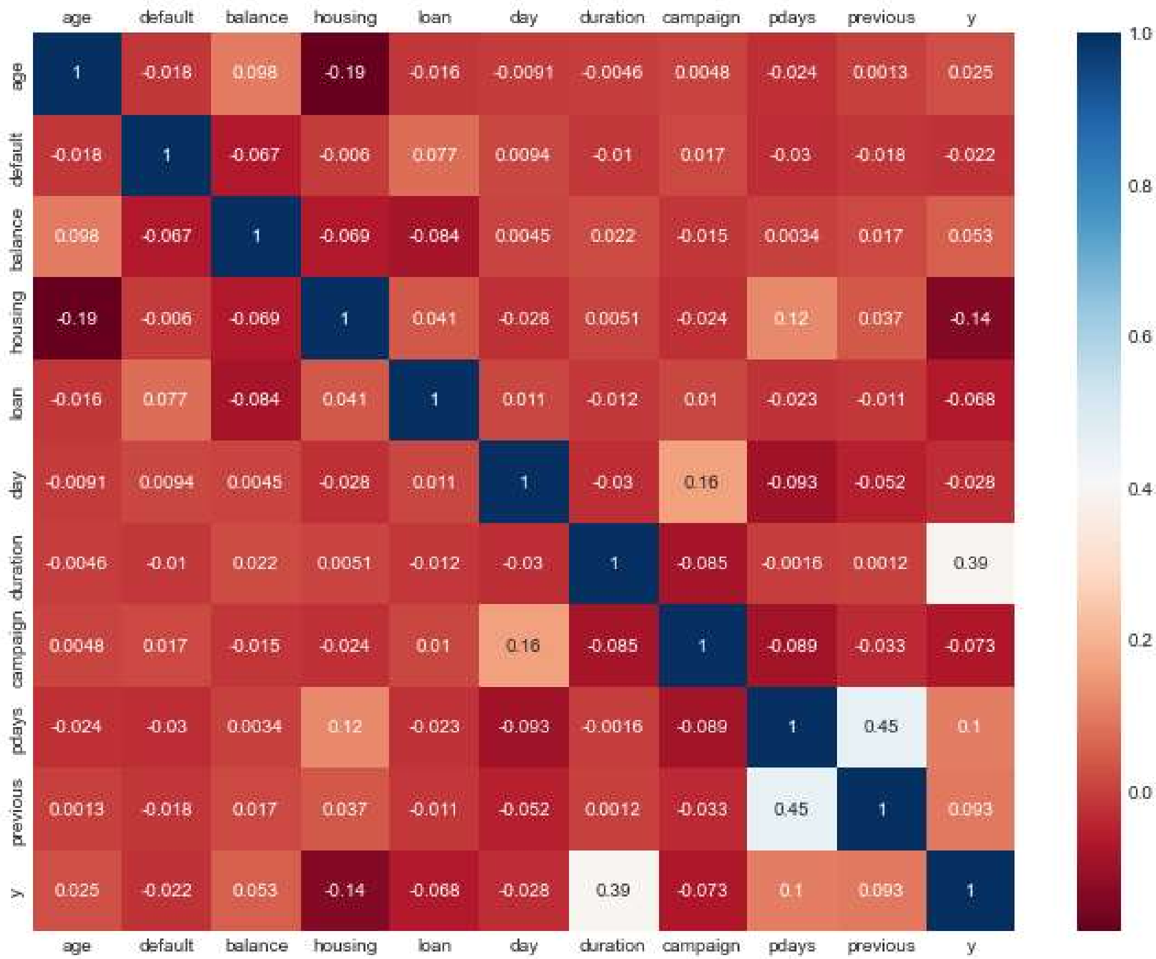
In [27]:
```
df.corr().round(2)
```

Out[27]:

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous
age	1.00	-0.02	0.10	-0.19	-0.02	-0.01	-0.00	0.00	-0.02	0.00
default	-0.02	1.00	-0.07	-0.01	0.08	0.01	-0.01	0.02	-0.03	-0.02
balance	0.10	-0.07	1.00	-0.07	-0.08	0.00	0.02	-0.01	0.00	0.02
housing	-0.19	-0.01	-0.07	1.00	0.04	-0.03	0.01	-0.02	0.12	0.04
loan	-0.02	0.08	-0.08	0.04	1.00	0.01	-0.01	0.01	-0.02	-0.01
day	-0.01	0.01	0.00	-0.03	0.01	1.00	-0.03	0.16	-0.09	-0.05
duration	-0.00	-0.01	0.02	0.01	-0.01	-0.03	1.00	-0.08	-0.00	0.00
campaign	0.00	0.02	-0.01	-0.02	0.01	0.16	-0.08	1.00	-0.09	-0.03
pdays	-0.02	-0.03	0.00	0.12	-0.02	-0.09	-0.00	-0.09	1.00	0.45
previous	0.00	-0.02	0.02	0.04	-0.01	-0.05	0.00	-0.03	0.45	1.00
y	0.03	-0.02	0.05	-0.14	-0.07	-0.03	0.39	-0.07	0.10	0.09

In [28]:
```
corr = df.corr()
plt.rcParams['xtick.top'] = plt.rcParams['xtick.labeltop'] = True
plt.figure(figsize=(12, 9))
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns,annot=Tr
```

Out[28]: <AxesSubplot:>



Most positively correlated variables:

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)

Principle Component Analysis (PCA)

PCA performed with all numeric variables

In [29]:
```
#Importing sklearn preprocesing package to use the MinMaxScaler
#MinMaxScaler is a rescaling function that translates all values into a range
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
scale.fit(bank_num)
scaled = scale.transform(bank_num)
```

In [30]:
```
#Importing sklearn decomposition package to use PCA
#PCA is a dimensionality reduction methodology that transforms all dimensions
#of the intial variable so that the first new component retains most of the v
from sklearn.decomposition import PCA
pca=PCA(n_components=7)
pca.fit(scaled)
transformed=pca.transform(scaled)
```

In [31]:
```
#Showing the original dimensionality of the data set
print("Dataframe original dimension: ", bank_num.shape)

#Showing the new dimensionality of the data set
print("Dataframe dimension after PCA: ", transformed.shape)
```

```
Dataframe original dimension:  (45211, 7)
Dataframe dimension after PCA:  (45211, 7)
```

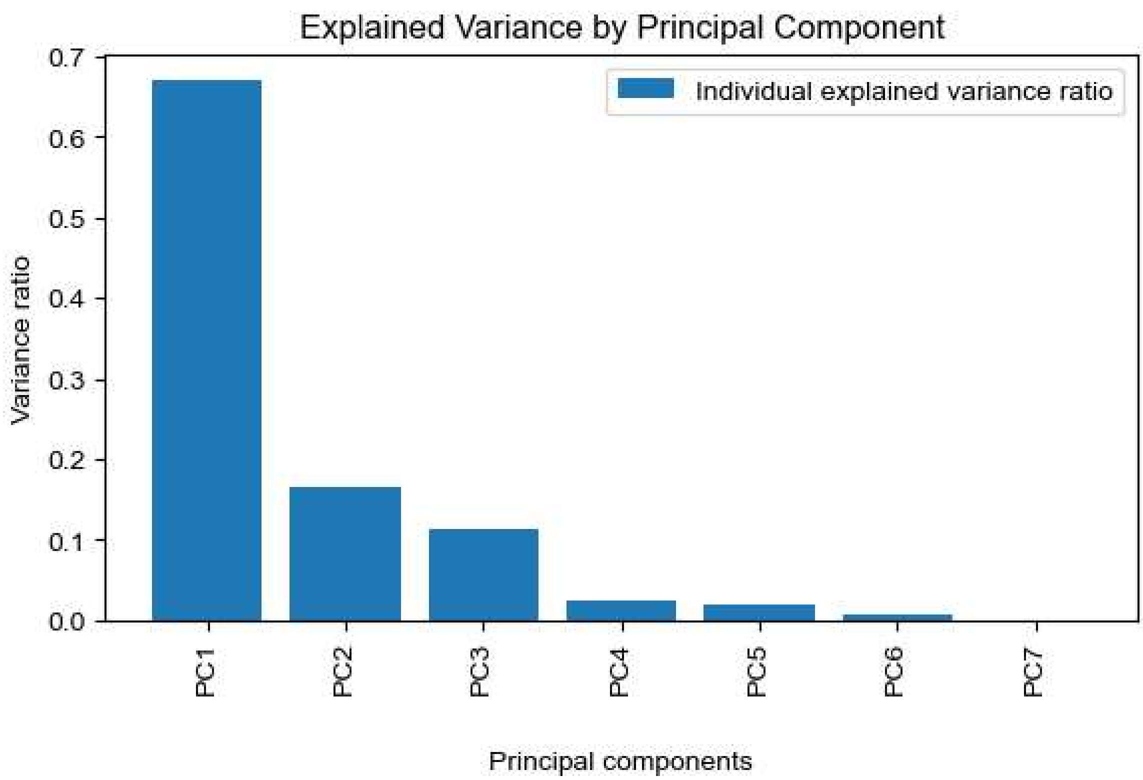
In [32]:
```
#Finding the explained variance ratio that corresponds to each new component
explained_variance=pca.explained_variance_ratio_
explained_variance
```

Out[32]:
```
array([6.69678811e-01, 1.65311078e-01, 1.13259502e-01, 2.46591163e-02,
       2.00510276e-02, 6.55843260e-03, 4.82031665e-04])
```

In [33]:

```
#Showing in a bar chart the explained variance, PC1 (index 0) accounts for 66
with plt.style.context('default'):
    plt.figure(figsize=(6, 4))
    plt.bar(range(7),explained_variance, align='center', label='Individual ex
    plt.xlabel('Principal components',labelpad=20)
    plt.ylabel('Variance ratio')
    plt.title('Explained Variance by Principal Component')
    plt.legend()
    plt.tight_layout()

    labels=['PC1', 'PC2', 'PC3','PC4','PC5','PC6','PC7']
    plt.xticks(range(7),labels, rotation='vertical')
```



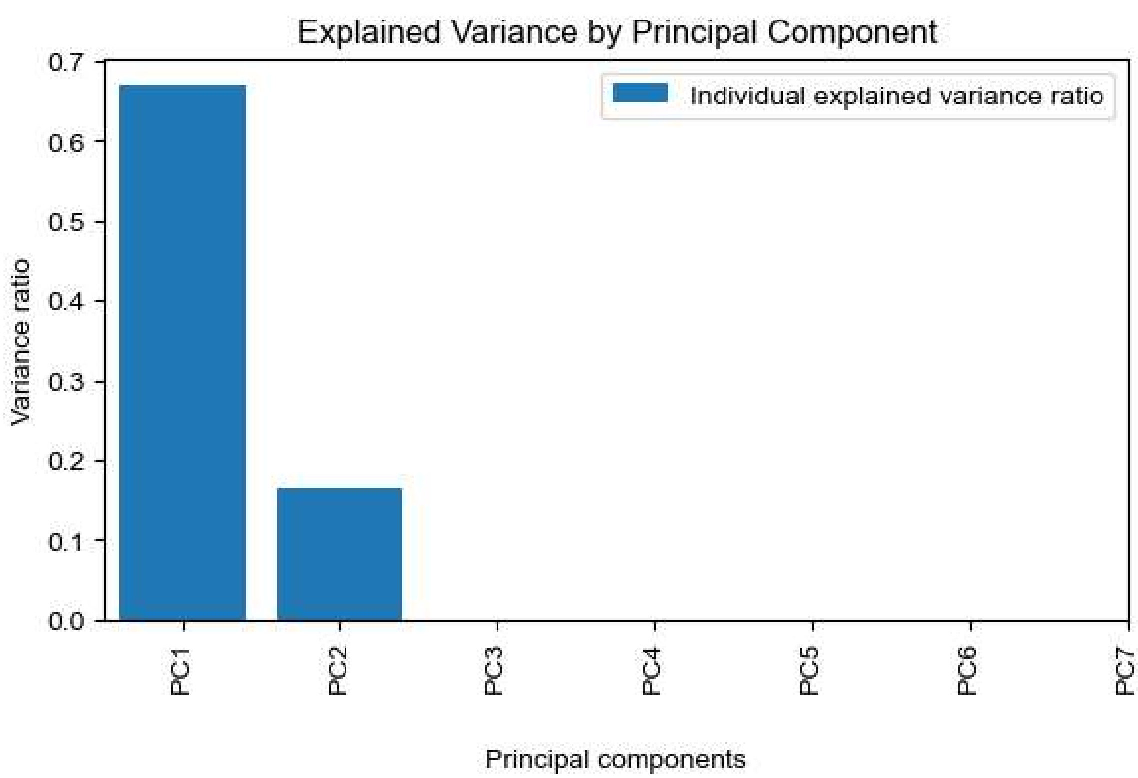
In [34]:
```
#Importing sklearn decomposition package to use PCA
#PCA is a dimensionality reduction methodology that transforms all dimensions
#of the intial variable so that the first new component retains most of the v
pca1=PCA(n_components=2)
pca1.fit(scaled)
transformed=pca1.transform(scaled)
```

In [35]:
```
#Finding the explained variance ratio that corresponds to each new component
explained_variance1=pca1.explained_variance_ratio_
explained_variance1
```

Out[35]:
```
array([0.66967881, 0.16531108])
```

In [36]:
```
#Showing in a bar chart the explained variance, PC1 (index 0) accounts for 66
with plt.style.context('default'):
    plt.figure(figsize=(6, 4))
    plt.bar(range(2),explained_variance1, align='center', label='Individual e
    plt.xlabel('Principal components',labelpad=20)
    plt.ylabel('Variance ratio')
    plt.title('Explained Variance by Principal Component')
    plt.legend()
    plt.tight_layout()

    labels=['PC1', 'PC2', 'PC3','PC4','PC5','PC6','PC7']
    plt.xticks(range(7),labels, rotation='vertical')
```



PCA performed with 2 variables

In [37]:
```
#In the previous sections of this review of the dataset it was found that dur
#the strongest impact on the target variable
#Therefore, the variance information in this variables may be transformed int
#so that only one dimension is left with most of the information

#Creating a subset containing duration and pdays
pca_red=['duration','pdays']
df_red2 = df[pca_red]
df_red2.head()
```

Out[37]:

	duration	pdays
0	261	-1
1	151	-1
2	76	-1
3	92	-1
4	198	-1

In [38]:
```
# Finding the corralation between the 2 variables
df_red2.corr()
```

Out[38]:

	duration	pdays
duration	1.000000	-0.001565
pdays	-0.001565	1.000000

In [39]:
```
#Finding the covariance matrix
df_red2.cov()
```

Out[39]:

	duration	pdays
duration	66320.574090	-40.349073
pdays	-40.349073	10025.765774

In [40]:
```
#Showing the original dimensions of the subset
df_red2.shape
```

Out[40]: (45211, 2)

In [41]:
```
#If not imported before, sklearn preprocesing needs to be imported for scalin
#Importing sklearn preprocesing package to use the MinMaxScaler
#MinMaxScaler is a rescaling function that translates all values into a range

scale=MinMaxScaler()
scale.fit(df_red2)
scaled = scale.transform(df_red2)
```

In [42]:
```
#If not imported before, sklearn preprocesing needs to be imported for scalin

#Importing sklearn decomposition package to use PCA
#PCA is a dimensionality reduction methodology that transforms all dimensions
#of the intial variable so that the first new component retains most of the v


from sklearn.decomposition import PCA
pca2=PCA(n_components=2)
pca2.fit(scaled)
transformed=pca2.transform(scaled)
```

In [43]:
```
x = transformed[:,1]
y = transformed[:,0]
```

In [44]:
```
#Showing the diemsnions of the transformed subset
transformed.shape
```

Out[44]:
```
(45211, 2)
```

In [45]:
```
#Finding the explained variance ratio that corresponds to each new component
explained_variance2=pca2.explained_variance_ratio_
explained_variance2
```

Out[45]:
```
array([0.82784027, 0.17215973])
```

In [46]:

```
#Showing in a bar chart the explained variance, PC1 (index 0) accounts for 82
with plt.style.context('default'):
    plt.figure(figsize=(6, 4))
    plt.bar(range(2),explained_variance2, align='center', label='Individual e
    plt.xlabel('Principal components',labelpad=15)
    plt.ylabel('Explained variance ratio')
    plt.title('Explained Variance by Principal Component')
    plt.legend()
    plt.tight_layout()

    labels=['PC1', 'PC2']
    plt.xticks(range(2),labels, rotation='vertical')
```