Pre-Modeling: Data Preprocessing and Feature Exploration in Python

GOAL

- Goal:

  Pre-modeling/modeling 80%/20% of work Show the importance of data preprocessing, feature exploration, and feature engineering on model performace *Go over a few effective pre-modeling steps

- Python libraries:
  - Numpy
  - Pandas
  - Sci-kit learn
  - Matplotlib

*Source of 'bank marketing' dataset: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

BANK MARKETING

Data Set 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, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

Attribute Information:

Input variables:

bank client data:

1. age: (numeric)
2. job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3. marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4. education (categorical: basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown
5. default: has credit in default? (categorical: 'no','yes','unknown')
6. housing: has housing loan? (categorical: 'no','yes','unknown')
7. loan: has personal loan? (categorical: 'no','yes','unknown') ##### related with the last contact of the current campaign:

8. contact: contact communication type (categorical: 'cellular','telephone')
9. month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10. day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11. duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, 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 only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model. ###### other attributes:
12. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. 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)
14. previous: number of contacts performed before this campaign and for this client (numeric)
15. poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

1. Introduce the Data
2. Basic Data Cleaning
    A. Dealing with data types
    B. Handling missing data
3. More Data Exploration
    A. Outlier detection
    B. Plotting distributions
4. Feature Engineering
    A. Relations between features
    B. Dimensionality reduction using PCA

PART 1: Introduce The Data

In [1]:
```
import import_ipynb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df=pd.read_csv(r"C:/Users/devro/Desktop/Jupyter/Recognizance21/bank.csv",sep=
```

In [3]:
```
df
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct

	age	job	marital	education	default	balance	housing	loan	contact	day	month
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may
...	...	...	...	...	...	...	...	...	...	...	...
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	jul
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9	may
4518	57	technician	married	secondary	no	295	no	no	cellular	19	aug
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6	feb
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3	apr

4521 rows × 17 columns

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

Out[4]:

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

In [5]:
```
df.tail()
```

Out[5]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	jul
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9	may
4518	57	technician	married	secondary	no	295	no	no	cellular	19	aug
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6	feb
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3	apr

Part 2: Basic data cleaning

A. Dealing with data types

- There are three main data types:
  - Numeric, e.g. income, age
  - Categorical, e.g. gender, nationality
  - Ordinal, e.g. low/medium/high
- Models can only handle numeric features
- Must convert categorical and ordinal features into numeric features
  - Create dummy features
  - Transform a categorical feature into a set of dummy features, each representing a unique category
  - In the set of dummy features, 1 indicates that the observation belongs to that category

In [6]:
```
for col_names in df.columns:
    if df[col_names].dtypes=='object':
        unique_cat=len(df[col_names].unique())
        print("Feature '{col_name}' has {unique_val} unique
categories".format(
        col_name=col_names,unique_val=unique_cat))
```

```
Feature 'job' has 12 unique categories
Feature 'marital' has 3 unique categories
Feature 'education' has 4 unique categories
Feature 'default' has 2 unique categories
Feature 'housing' has 2 unique categories
Feature 'loan' has 2 unique categories
Feature 'contact' has 3 unique categories
Feature 'month' has 12 unique categories
Feature 'poutcome' has 4 unique categories
Feature 'y' has 2 unique categories
```

In [7]:
```
def replace_marital(val):
    if val=="single":
        return 0
    else:
        return 1
df["marital"]=df["marital"].apply(replace_marital)
```

In [8]:
```
df.head()
```

Out[8]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	dura
0	30	unemployed	1	primary	no	1787	no	no	cellular	19	oct	
1	33	services	1	secondary	no	4789	yes	yes	cellular	11	may	
2	35	management	0	tertiary	no	1350	yes	no	cellular	16	apr	

	age	job	marital	education	default	balance	housing	loan	contact	day	month	dura
3	30	management	1	tertiary	no	1476	yes	yes	unknown	3	jun	
4	59	blue-collar	1	secondary	no	0	yes	no	unknown	5	may	

In [9]:
```
df["housing"]=df["housing"].map({
    "no":0,
    "yes":1
})
```

In [10]:
```
df.head()
```

Out[10]:

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

In [11]:
```
df["loan"]=df["loan"].replace({
    "no":0,
    "yes":1
})
```

In [12]:
```
df.head()
```

Out[12]:

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

In [13]:
```
df["job"].unique()
```

Out[13]:
```
array(['unemployed', 'services', 'management', 'blue-collar',
       'self-employed', 'technician', 'entrepreneur', 'admin.', 'student',
       'housemaid', 'retired', 'unknown'], dtype=object)
```

In [14]:
```
df["job"].value_counts().sort_values(ascending=False)
```

Out[14]:
```
management       969
blue-collar      946
technician       768
admin.           478
services         417
retired          230
self-employed    183
entrepreneur     168
unemployed       128
housemaid        112
student           84
unknown           38
Name: job, dtype: int64
```

In [15]:
```
df["job"]=df["job"].map({
    'unknown':np.nan,
    'unemployed':0,
    'services':1,
    'management':2,
    'blue-collar':3,
    'self-employed':4,
    'technician':5,
    'entrepreneur':6,
    'admin.':7,
    'student':8,
    'housemaid':9,
    'retired':10,
})
```

In [16]:
```
df.head()
```

Out[16]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30	0.0	1	primary	no	1787	0	0	cellular	19	oct	79	
1	33	1.0	1	secondary	no	4789	1	1	cellular	11	may	220	
2	35	2.0	0	tertiary	no	1350	1	0	cellular	16	apr	185	
3	30	2.0	1	tertiary	no	1476	1	1	unknown	3	jun	199	
4	59	3.0	1	secondary	no	0	1	0	unknown	5	may	226	

In [17]:
```
df.tail()
```

Out[17]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
4516	33	1.0	1	secondary	no	-333	1	0	cellular	30	jul	329

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
4517	57	4.0	1	tertiary	yes	-3313	1	1	unknown	9	may	153
4518	57	5.0	1	secondary	no	295	0	0	cellular	19	aug	151
4519	28	3.0	1	secondary	no	1137	0	0	cellular	6	feb	129
4520	44	6.0	0	tertiary	no	1136	1	1	cellular	3	apr	345

In [18]:
```
df["education"].unique()
```

Out[18]:
```
array(['primary', 'secondary', 'tertiary', 'unknown'], dtype=object)
```

In [19]:
```
df["education"].replace({
    'unknown':np.nan,
    'primary':0,
    'secondary':1,
    'tertiary':2
},inplace=True)
```

In [20]:
```
df.head()
```

Out[20]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30	0.0	1	0.0	no	1787	0	0	cellular	19	oct	79	
1	33	1.0	1	1.0	no	4789	1	1	cellular	11	may	220	
2	35	2.0	0	2.0	no	1350	1	0	cellular	16	apr	185	
3	30	2.0	1	2.0	no	1476	1	1	unknown	3	jun	199	
4	59	3.0	1	1.0	no	0	1	0	unknown	5	may	226	

In [21]:
```
df["default"]=[0 if x=="no" else 1 for x in df["default"]]
```

In [22]:
```
df.head()
```

Out[22]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30	0.0	1	0.0	0	1787	0	0	cellular	19	oct	79	
1	33	1.0	1	1.0	0	4789	1	1	cellular	11	may	220	
2	35	2.0	0	2.0	0	1350	1	0	cellular	16	apr	185	
3	30	2.0	1	2.0	0	1476	1	1	unknown	3	jun	199	
4	59	3.0	1	1.0	0	0	1	0	unknown	5	may	226	

In [23]:
```
x=df.drop("default",1)
x.head(10)
```

Out[23]:

	age	job	marital	education	balance	housing	loan	contact	day	month	duration	campaign
0	30	0.0	1	0.0	1787	0	0	cellular	19	oct	79	1
1	33	1.0	1	1.0	4789	1	1	cellular	11	may	220	1
2	35	2.0	0	2.0	1350	1	0	cellular	16	apr	185	1
3	30	2.0	1	2.0	1476	1	1	unknown	3	jun	199	4
4	59	3.0	1	1.0	0	1	0	unknown	5	may	226	1
5	35	2.0	0	2.0	747	0	0	cellular	23	feb	141	2
6	36	4.0	1	2.0	307	1	0	cellular	14	may	341	1
7	39	5.0	1	1.0	147	1	0	cellular	6	may	151	2
8	41	6.0	1	2.0	221	1	0	unknown	14	may	57	2
9	43	1.0	1	0.0	-88	1	1	cellular	17	apr	313	1

In [24]:
```
y=df.default
y.tail(10)
```

Out[24]:
```
4511    0
4512    0
4513    0
4514    0
4515    0
4516    0
4517    1
4518    0
4519    0
4520    0
Name: default, dtype: int64
```

In [25]:
```
df["balance"].min()
```

Out[25]:
```
-3313
```

In [26]:
```
df["balance"].max()
```

Out[26]:
```
71188
```

In [27]:
```
df["balance"].mean()
#df["job"].value_counts().plot(kind='bar',figsize=(10,10))
```

Out[27]:
```
1422.6578190665782
```

In [28]:
```

```

```
df["balance"]=df["balance"].apply(lambda v:(v-
df["balance"].min())/(df["balance"].max()-df["balance"].min()))
```

In [29]:
```
df.head()
```

Out[29]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30	0.0	1	0.0	0	0.068455	0	0	cellular	19	oct	79	
1	33	1.0	1	1.0	0	0.108750	1	1	cellular	11	may	220	
2	35	2.0	0	2.0	0	0.062590	1	0	cellular	16	apr	185	
3	30	2.0	1	2.0	0	0.064281	1	1	unknown	3	jun	199	
4	59	3.0	1	1.0	0	0.044469	1	0	unknown	5	may	226	

In [30]:
```
df.tail()
```

Out[30]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
4516	33	1.0	1	1.0	0	0.039999	1	0	cellular	30	jul	329
4517	57	4.0	1	2.0	1	0.000000	1	1	unknown	9	may	153
4518	57	5.0	1	1.0	0	0.048429	0	0	cellular	19	aug	151
4519	28	3.0	1	1.0	0	0.059731	0	0	cellular	6	feb	129
4520	44	6.0	0	2.0	0	0.059717	1	1	cellular	3	apr	345

In [31]:
```
df.contact.unique()
#df.contact.value_counts().plot(kind='bar',figsize=(10,10))
```

Out[31]:
```
array(['cellular', 'unknown', 'telephone'], dtype=object)
```

In [32]:
```
df.contact.replace({
    'unknown':np.nan,
    'cellular':1,
    'telephone':2
},inplace=True)
```

In [33]:
```
df.head()
```

Out[33]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	car
0	30	0.0	1	0.0	0	0.068455	0	0	1.0	19	oct	79	
1	33	1.0	1	1.0	0	0.108750	1	1	1.0	11	may	220	

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	car
2	35	2.0	0	2.0	0	0.062590	1	0	1.0	16	apr	185	
3	30	2.0	1	2.0	0	0.064281	1	1	NaN	3	jun	199	
4	59	3.0	1	1.0	0	0.044469	1	0	NaN	5	may	226	

In [34]:
```
df.tail()
```

Out[34]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
4516	33	1.0	1	1.0	0	0.039999	1	0	1.0	30	jul	329
4517	57	4.0	1	2.0	1	0.000000	1	1	NaN	9	may	153
4518	57	5.0	1	1.0	0	0.048429	0	0	1.0	19	aug	151
4519	28	3.0	1	1.0	0	0.059731	0	0	1.0	6	feb	129
4520	44	6.0	0	2.0	0	0.059717	1	1	1.0	3	apr	345

In [35]:
```
df.month.unique()
```

Out[35]:
```
array(['oct', 'may', 'apr', 'jun', 'feb', 'aug', 'jan', 'jul', 'nov',
       'sep', 'mar', 'dec'], dtype=object)
```

In [36]:
```
df["month"]=df["month"].map({
    "jan":1,
    "feb":2,
    "mar":3,
    "apr":4,
    "may":5,
    "jun":6,
    "jul":7,
    "aug":8,
    "sep":9,
    "oct":10,
    "nov":11,
    "dec":12
}.get)
```

In [37]:
```
df.poutcome.unique()
```

Out[37]:
```
array(['unknown', 'failure', 'other', 'success'], dtype=object)
```

In [38]:
```
df.poutcome.replace({
```

```
    'unknown':np.nan,
    'failure':0,
    'other':1,
    'success':2
},inplace=True)
```

In [39]:
```
df.head()
```

Out[39]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30	0.0	1	0.0	0	0.068455	0	0	1.0	19	10	79	
1	33	1.0	1	1.0	0	0.108750	1	1	1.0	11	5	220	
2	35	2.0	0	2.0	0	0.062590	1	0	1.0	16	4	185	
3	30	2.0	1	2.0	0	0.064281	1	1	NaN	3	6	199	
4	59	3.0	1	1.0	0	0.044469	1	0	NaN	5	5	226	

In [40]:
```
df.poutcome.unique()
```

Out[40]:
```
array([nan,  0.,  1.,  2.])
```

In [41]:
```
df.y.replace({
    'no':0,
    'yes':1
},inplace=True)
```

In [42]:
```
df.head()
```

Out[42]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30	0.0	1	0.0	0	0.068455	0	0	1.0	19	10	79	
1	33	1.0	1	1.0	0	0.108750	1	1	1.0	11	5	220	
2	35	2.0	0	2.0	0	0.062590	1	0	1.0	16	4	185	
3	30	2.0	1	2.0	0	0.064281	1	1	NaN	3	6	199	
4	59	3.0	1	1.0	0	0.044469	1	0	NaN	5	5	226	

In [43]:
```
df.tail()
```

Out[43]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
4516	33	1.0	1	1.0	0	0.039999	1	0	1.0	30	7	329

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
4517	57	4.0	1	2.0	1	0.000000	1	1	NaN	9	5	153
4518	57	5.0	1	1.0	0	0.048429	0	0	1.0	19	8	151
4519	28	3.0	1	1.0	0	0.059731	0	0	1.0	6	2	129
4520	44	6.0	0	2.0	0	0.059717	1	1	1.0	3	4	345

In [44]:
```
df.duration=df.duration.apply(lambda v:(v-
df.duration.min())/(df.duration.max()-df.duration.min()))
```

In [45]:
```
df.head()
```

Out[45]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	car
0	30	0.0	1	0.0	0	0.068455	0	0	1.0	19	10	0.024826	
1	33	1.0	1	1.0	0	0.108750	1	1	1.0	11	5	0.071500	
2	35	2.0	0	2.0	0	0.062590	1	0	1.0	16	4	0.059914	
3	30	2.0	1	2.0	0	0.064281	1	1	NaN	3	6	0.064548	
4	59	3.0	1	1.0	0	0.044469	1	0	NaN	5	5	0.073486	

In [46]:
```
df.tail()
```

Out[46]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
4516	33	1.0	1	1.0	0	0.039999	1	0	1.0	30	7	0.107580
4517	57	4.0	1	2.0	1	0.000000	1	1	NaN	9	5	0.049321
4518	57	5.0	1	1.0	0	0.048429	0	0	1.0	19	8	0.048659
4519	28	3.0	1	1.0	0	0.059731	0	0	1.0	6	2	0.041377
4520	44	6.0	0	2.0	0	0.059717	1	1	1.0	3	4	0.112877

In [47]:
```
df.pdays=df.pdays.apply(lambda v:(v-df.pdays.min())/(df.pdays.max()-
df.pdays.min()))
```

In [48]:
```
df.head()
```

Out[48]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	car
0	30	0.0	1	0.0	0	0.068455	0	0	1.0	19	10	0.024826	
1	33	1.0	1	1.0	0	0.108750	1	1	1.0	11	5	0.071500	

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
2	35	2.0	0	2.0	0	0.062590	1	0	1.0	16	4	0.059914	
3	30	2.0	1	2.0	0	0.064281	1	1	NaN	3	6	0.064548	
4	59	3.0	1	1.0	0	0.044469	1	0	NaN	5	5	0.073486	

In [49]:
```
df.describe()
```

Out[49]:

	age	job	marital	education	default	balance	housing	
count	4521.000000	4483.000000	4521.000000	4334.000000	4521.000000	4521.000000	4521.000000	452
mean	41.170095	4.037252	0.735457	1.155053	0.016810	0.063565	0.566025	
std	10.576211	2.534139	0.441138	0.666325	0.128575	0.040397	0.495676	
min	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	33.000000	2.000000	0.000000	1.000000	0.000000	0.045395	0.000000	
50%	39.000000	3.000000	1.000000	1.000000	0.000000	0.050429	1.000000	
75%	49.000000	5.000000	1.000000	2.000000	0.000000	0.064335	1.000000	
max	87.000000	10.000000	1.000000	2.000000	1.000000	1.000000	1.000000	

In [50]:
```
df.to_csv(r"C:/Users/devro/Desktop/Jupyter/Recognizance21/preprocessed.csv",i
```

In [51]:
```
new_df=pd.read_csv(r"C:/Users/devro/Desktop/Jupyter/Recognizance21/preprocess
```

In [52]:
```
new_df.head()
```

Out[52]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30	0.0	1	0.0	0	0.068455	0	0	1.0	19	10	0.024826	
1	33	1.0	1	1.0	0	0.108750	1	1	1.0	11	5	0.071500	
2	35	2.0	0	2.0	0	0.062590	1	0	1.0	16	4	0.059914	
3	30	2.0	1	2.0	0	0.064281	1	1	NaN	3	6	0.064548	
4	59	3.0	1	1.0	0	0.044469	1	0	NaN	5	5	0.073486	

B. Handling missing data

- Models can not handle missing data
- Simplest solution

    - Remove observations/features that have missing data
- But, removing missing data can introduce a lot of issues
    - Data is randomly missing: potentially lose a lot of your data
    - Data is non-randomly missing: in addition to losing data, you are also introducing potential biases
    - Usually, this is a poor solution
- An alternative solution is to use imputation
    - Replace missing value with another value
    - Strategies: mean, median, highest frequency value of given feature

In [53]:
```
# Now check to see if you still have missing data
new_df.isnull().sum().sort_values(ascending=False)
```

Out[53]:
```
poutcome     3705
contact      1324
education     187
job            38
loan            0
marital         0
default         0
balance         0
housing         0
y               0
day             0
month           0
duration        0
campaign        0
pdays           0
previous        0
age             0
dtype: int64
```

In [54]:
```
# Impute missing values using Imputer in sklearn.preprocessing
from sklearn.impute import SimpleImputer


imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(new_df)
new_df = pd.DataFrame(data=imp.transform(new_df) , columns=new_df.columns)
```

In [55]:
```
# Now check again to see if you still have missing data
new_df.isnull().sum().sort_values().head()
```

Out[55]:
```
age         0
previous    0
pdays       0
campaign    0
duration    0
dtype: int64
```

In [56]:
```
new_df
```

Out[56]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
0	30.0	0.0	1.0	0.0	0.0	0.068455	0.0	0.0	1.0	19.0	10.0	0.024826
1	33.0	1.0	1.0	1.0	0.0	0.108750	1.0	1.0	1.0	11.0	5.0	0.071500
2	35.0	2.0	0.0	2.0	0.0	0.062590	1.0	0.0	1.0	16.0	4.0	0.059914
3	30.0	2.0	1.0	2.0	0.0	0.064281	1.0	1.0	1.0	3.0	6.0	0.064548
4	59.0	3.0	1.0	1.0	0.0	0.044469	1.0	0.0	1.0	5.0	5.0	0.073486
...	...	...	...	...	...	...	...	...	...	...	...	...
4516	33.0	1.0	1.0	1.0	0.0	0.039999	1.0	0.0	1.0	30.0	7.0	0.107580
4517	57.0	4.0	1.0	2.0	1.0	0.000000	1.0	1.0	1.0	9.0	5.0	0.049321
4518	57.0	5.0	1.0	1.0	0.0	0.048429	0.0	0.0	1.0	19.0	8.0	0.048659
4519	28.0	3.0	1.0	1.0	0.0	0.059731	0.0	0.0	1.0	6.0	2.0	0.041377
4520	44.0	6.0	0.0	2.0	0.0	0.059717	1.0	1.0	1.0	3.0	4.0	0.112877

4521 rows × 17 columns

Part 3: More Data Exploration

- A large portion of the pre-modeling and modeling workflow can be generalized and automated
- But understanding the problem, domain, and data is extremely important for building high performing models
- This section covers some tools used for exploring your data to make smarter decisions

A. Outlier detection

- An outlier is an observation that deviates drastically from other observations in a dataset
- Occurrence:
  - Natural, e.g. Mark Zuckerberg's income
  - Error, e.g. human weight of 700 kg due to mistyping extra 0
- Why are they problematic?
  - Naturally occuring:
    - Not necessarily problematic
    - But can skew your model by affecting the slope (see image below)
  - Error
    - Indicative of data quality issues
    - Treat in the same way as a missing value, i.e. use imputation
- Many, many approaches for detecting outliers; we will discuss two of these:
  - Tukey IQR
  - Kernel density estimatation

In [57]:
```
from IPython.display import Image
Image(filename=r'C:\Users\devro\Desktop\Jupyter\Recognizance21\outliers.jpg')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-57-4b52b439abae> in <module>
      1 from IPython.display import Image
----> 2 Image(filename=r'C:\Users\devro\Desktop\Jupyter\Recognizance21\outliers.jpg')

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in __init__(self, dat
a, url, filename, format, embed, width, height, retina, unconfined, metadata)
   1223         self.unconfined = unconfined
   1224         super(Image, self).__init__(data=data, url=url, filename=filename,
-> 1225                 metadata=metadata)
   1226
   1227         if self.width is None and self.metadata.get('width', {}):

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in __init__(self, dat
a, url, filename, metadata)
    628             self.metadata = {}
    629
--> 630         self.reload()
    631         self._check_data()
    632

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in reload(self)
   1254         """Reload the raw data from file or URL."""
   1255         if self.embed:
-> 1256             super(Image,self).reload()
   1257             if self.retina:
   1258                 self._retina_shape()

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in reload(self)
    653         """Reload the raw data from file or URL."""
    654         if self.filename is not None:
--> 655             with open(self.filename, self._read_flags) as f:
    656                 self.data = f.read()
    657         elif self.url is not None:

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\devro\\Desktop\\Jupy
ter\\Recognizance21\\outliers.jpg'
```

Outlier detection - Tukey IQR

- Identifies extreme values in data
- Outliers are defined as:
  - Values below Q1-1.5(Q3-Q1) or above Q3+1.5(Q3-Q1)
- Standard deviation from the mean is another common method to detect extreme values
  - But it can be problematic:
    - Assumes normality
    - Sensitive to very extreme values

In [ ]:
```
from IPython.display import Image
Image(filename=r'C:\Users\abhir\OneDrive\Desktop\bank\tukeyiqr.jpg')
```

In [58]:
```
new_df.describe()
```

Out[58]:

	age	job	marital	education	default	balance	housing	
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	452
mean	41.170095	4.028534	0.735457	1.148640	0.016810	0.063565	0.566025	(
std	10.576211	2.525241	0.441138	0.653126	0.128575	0.040397	0.495676	(
min	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(
25%	33.000000	2.000000	0.000000	1.000000	0.000000	0.045395	0.000000	(
50%	39.000000	3.000000	1.000000	1.000000	0.000000	0.050429	1.000000	(
75%	49.000000	5.000000	1.000000	2.000000	0.000000	0.064335	1.000000	(
max	87.000000	10.000000	1.000000	2.000000	1.000000	1.000000	1.000000	

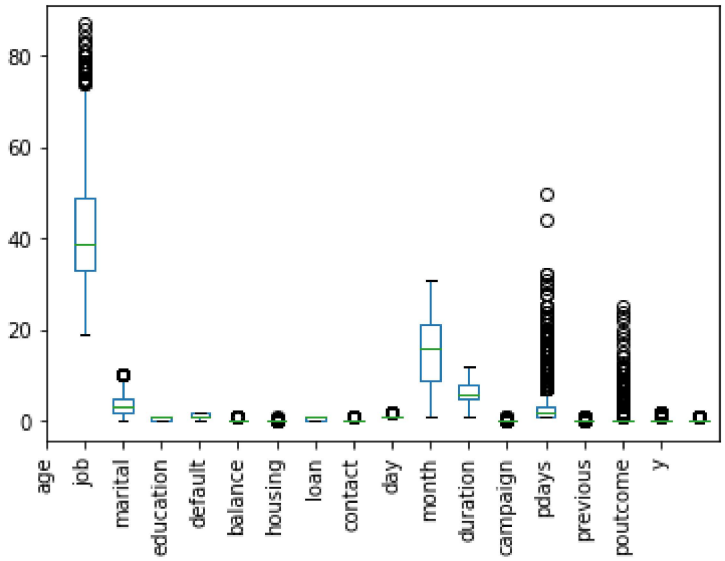
In [59]:
```
new_df.age.value_counts()
```

Out[59]:
```
34.0    231
32.0    224
31.0    199
36.0    188
33.0    186
       ...
76.0      2
86.0      1
81.0      1
84.0      1
87.0      1
Name: age, Length: 67, dtype: int64
```

In [60]:
```
new_df.plot.box()
plt.xticks(list(range(len(new_df.columns))),new_df.columns,rotation="vertical
```

Out[60]:
```
([<matplotlib.axis.XTick at 0x12a9d1ae470>,
  <matplotlib.axis.XTick at 0x12a9d1ae048>,
  <matplotlib.axis.XTick at 0x12a9d1a9c50>,
  <matplotlib.axis.XTick at 0x12a9d549e10>,
  <matplotlib.axis.XTick at 0x12a9d55b2e8>,
  <matplotlib.axis.XTick at 0x12a9d55b4e0>,
  <matplotlib.axis.XTick at 0x12a9d55b978>,
  <matplotlib.axis.XTick at 0x12a9d55be10>,
  <matplotlib.axis.XTick at 0x12a9d5632e8>,
  <matplotlib.axis.XTick at 0x12a9d563780>,
  <matplotlib.axis.XTick at 0x12a9d563c18>,
  <matplotlib.axis.XTick at 0x12a9d56b0f0>,
  <matplotlib.axis.XTick at 0x12a9d56b588>,
  <matplotlib.axis.XTick at 0x12a9d563828>,
  <matplotlib.axis.XTick at 0x12a9d55b710>,
  <matplotlib.axis.XTick at 0x12a9d56b198>,
  <matplotlib.axis.XTick at 0x12a9d56bdd8>],
 [Text(0, 0, 'age'),
  Text(1, 0, 'job'),
  Text(2, 0, 'marital'),
  Text(3, 0, 'education'),
  Text(4, 0, 'default'),
  Text(5, 0, 'balance'),
```

```
Text(6, 0, 'housing'),
Text(7, 0, 'loan'),
Text(8, 0, 'contact'),
Text(9, 0, 'day'),
Text(10, 0, 'month'),
Text(11, 0, 'duration'),
Text(12, 0, 'campaign'),
Text(13, 0, 'pdays'),
Text(14, 0, 'previous'),
Text(15, 0, 'poutcome'),
Text(16, 0, 'y')])
```



In [61]:

```
def find_outliers(x):
    q1 = np.percentile(x, 25)
    q3 = np.percentile(x, 75)
    iqr = q3-q1
    floor = q1 - 1.5*iqr
    ceiling = q3 + 1.5*iqr
    outlier_indices = list(x.index[(x < floor)|(x > ceiling)])
    outlier_values = list(x[outlier_indices])

    return outlier_indices, outlier_values
```

In [62]:

```
age_outliers_indices, age_outliers_values = find_outliers(new_df['age'])
print(np.sort(age_outliers_values))
```

```
[74. 74. 74. 75. 75. 75. 75. 75. 75. 76. 76. 77. 77. 77. 77. 77. 77. 78.
 78. 78. 79. 79. 79. 79. 80. 80. 80. 80. 80. 80. 81. 83. 83. 83. 83. 84.
 86. 87.]
```

In [63]:

```
campaign_outliers_indices, campaign_outliers_values =
find_outliers(new_df['campaign'])
print(np.sort(campaign_outliers_values))
```

```
[ 7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.
```

```
 7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.
 7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.
 7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.  7.
 7.  7.  7.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.
 8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.
 8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.  8.
 8.  8.  8.  8.  8.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.
 9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9.  9. 10.
10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10.
10. 10. 10. 10. 10. 10. 10. 10. 11. 11. 11. 11. 11. 11. 11. 11. 11.
11. 11. 11. 11. 11. 11. 11. 11. 11. 11. 11. 11. 12. 12. 12. 12. 12.
12. 12. 12. 12. 12. 12. 12. 12. 12. 12. 12. 12. 12. 12. 13. 13. 13.
13. 13. 13. 13. 13. 13. 13. 13. 13. 13. 13. 13. 13. 14. 14. 14. 14.
14. 14. 14. 14. 14. 14. 15. 15. 15. 15. 15. 15. 15. 15. 15. 16. 16.
16. 16. 16. 16. 16. 17. 17. 17. 17. 17. 17. 17. 18. 18. 18. 18. 18.
18. 19. 19. 19. 20. 20. 20. 21. 21. 22. 22. 23. 23. 24. 24. 24. 25.
25. 25. 28. 28. 28. 29. 30. 31. 32. 32. 44. 50.]
```

In [64]:

```
low=0.01
high=0.99
mid=0.50
new_df.quantile([low,mid,high])
```

Out[64]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
0.01	24.0	0.0	0.0	0.0	0.0	0.035457	0.0	0.0	1.0	2.0	1.0	0.002052
0.50	39.0	3.0	1.0	1.0	0.0	0.050429	1.0	0.0	1.0	16.0	6.0	0.059914
0.99	72.0	10.0	1.0	2.0	1.0	0.234998	1.0	1.0	2.0	31.0	11.0	0.415425

In [65]:

```
new_df.describe()
```

Out[65]:

	age	job	marital	education	default	balance	housing	
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	452
mean	41.170095	4.028534	0.735457	1.148640	0.016810	0.063565	0.566025	
std	10.576211	2.525241	0.441138	0.653126	0.128575	0.040397	0.495676	
min	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	33.000000	2.000000	0.000000	1.000000	0.000000	0.045395	0.000000	
50%	39.000000	3.000000	1.000000	1.000000	0.000000	0.050429	1.000000	
75%	49.000000	5.000000	1.000000	2.000000	0.000000	0.064335	1.000000	
max	87.000000	10.000000	1.000000	2.000000	1.000000	1.000000	1.000000	

In [66]:

```
qdf=new_df.quantile([low,mid,high])
```

In [67]:

```
qdf.age
```

Out[67]:
```
0.01    24.0
0.50    39.0
0.99    72.0
Name: age, dtype: float64
```

In [68]:
```
qdf.age[mid]
```

Out[68]:
```
39.0
```

In [69]:
```
new_df.age=new_df.age.apply(lambda v:v if qdf.age[low]<v<qdf.age[high] else
np.nan)
```

In [70]:
```
new_df.head()
```

Out[70]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30.0	0.0	1.0	0.0	0.0	0.068455	0.0	0.0	1.0	19.0	10.0	0.024826	
1	33.0	1.0	1.0	1.0	0.0	0.108750	1.0	1.0	1.0	11.0	5.0	0.071500	
2	35.0	2.0	0.0	2.0	0.0	0.062590	1.0	0.0	1.0	16.0	4.0	0.059914	
3	30.0	2.0	1.0	2.0	0.0	0.064281	1.0	1.0	1.0	3.0	6.0	0.064548	
4	59.0	3.0	1.0	1.0	0.0	0.044469	1.0	0.0	1.0	5.0	5.0	0.073486	

In [71]:
```
new_df.age.value_counts(ascending=False)
```

Out[71]:
```
34.0    231
32.0    224
31.0    199
36.0    188
33.0    186
35.0    180
37.0    161
38.0    159
30.0    150
40.0    142
42.0    141
41.0    135
39.0    130
46.0    119
43.0    115
48.0    114
45.0    112
49.0    112
47.0    108
44.0    105
28.0    103
50.0    101
29.0     97
27.0     94
53.0     94
57.0     91
51.0     91
55.0     90
52.0     86
```

```
58.0     85
26.0     77
56.0     74
59.0     71
54.0     71
60.0     47
25.0     44
61.0     16
66.0      9
63.0      8
64.0      7
62.0      7
70.0      7
65.0      6
69.0      6
71.0      6
67.0      5
68.0      2
Name: age, dtype: int64
```

In [72]:
```
new_df.age.dropna(inplace=True)
```

In [73]:
```
new_df.head()
```

Out[73]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
0	30.0	0.0	1.0	0.0	0.0	0.068455	0.0	0.0	1.0	19.0	10.0	0.024826	
1	33.0	1.0	1.0	1.0	0.0	0.108750	1.0	1.0	1.0	11.0	5.0	0.071500	
2	35.0	2.0	0.0	2.0	0.0	0.062590	1.0	0.0	1.0	16.0	4.0	0.059914	
3	30.0	2.0	1.0	2.0	0.0	0.064281	1.0	1.0	1.0	3.0	6.0	0.064548	
4	59.0	3.0	1.0	1.0	0.0	0.044469	1.0	0.0	1.0	5.0	5.0	0.073486	

In [74]:
```
new_df.age.value_counts()
```

Out[74]:
```
34.0    231
32.0    224
31.0    199
36.0    188
33.0    186
35.0    180
37.0    161
38.0    159
30.0    150
40.0    142
42.0    141
41.0    135
39.0    130
46.0    119
43.0    115
48.0    114
45.0    112
49.0    112
47.0    108
44.0    105
28.0    103
50.0    101
```

```
29.0     97
27.0     94
53.0     94
57.0     91
51.0     91
55.0     90
52.0     86
58.0     85
26.0     77
56.0     74
59.0     71
54.0     71
60.0     47
25.0     44
61.0     16
66.0      9
63.0      8
64.0      7
62.0      7
70.0      7
65.0      6
69.0      6
71.0      6
67.0      5
68.0      2
Name: age, dtype: int64
```

In [75]:
```
new_df.describe()
```

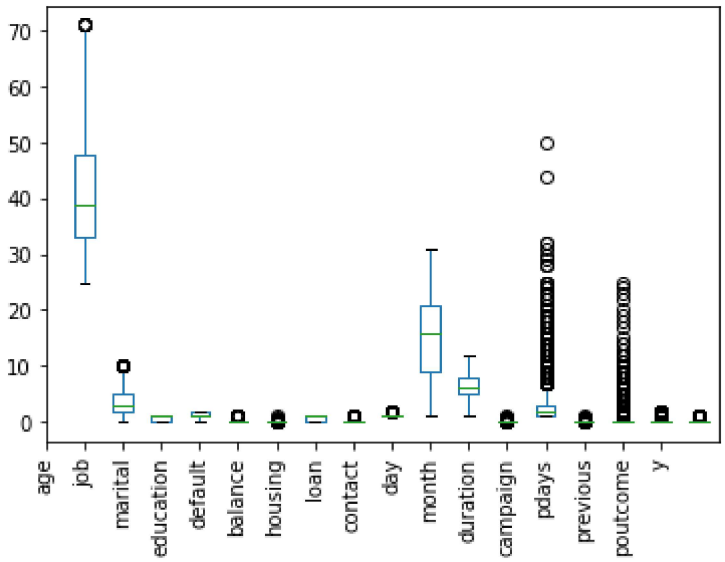
Out[75]:

	age	job	marital	education	default	balance	housing	
count	4406.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	452
mean	41.057876	4.028534	0.735457	1.148640	0.016810	0.063565	0.566025	(
std	9.751862	2.525241	0.441138	0.653126	0.128575	0.040397	0.495676	(
min	25.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(
25%	33.000000	2.000000	0.000000	1.000000	0.000000	0.045395	0.000000	(
50%	39.000000	3.000000	1.000000	1.000000	0.000000	0.050429	1.000000	(
75%	48.000000	5.000000	1.000000	2.000000	0.000000	0.064335	1.000000	(
max	71.000000	10.000000	1.000000	2.000000	1.000000	1.000000	1.000000	

In [76]:
```
new_df.plot.box()
plt.xticks(list(range(len(new_df.columns))),new_df.columns,rotation="vertical
```

Out[76]:
```
([<matplotlib.axis.XTick at 0x12a9d678358>,
  <matplotlib.axis.XTick at 0x12a9d67bef0>,
  <matplotlib.axis.XTick at 0x12a9d67bb38>,
  <matplotlib.axis.XTick at 0x12a9d7d90b8>,
  <matplotlib.axis.XTick at 0x12a9d7d9550>,
  <matplotlib.axis.XTick at 0x12a9d7d99e8>,
  <matplotlib.axis.XTick at 0x12a9d7d9e80>,
  <matplotlib.axis.XTick at 0x12a9d7e3358>,
  <matplotlib.axis.XTick at 0x12a9d7e37f0>,
```

```
 <matplotlib.axis.XTick at 0x12a9d7d9668>,
 <matplotlib.axis.XTick at 0x12a9d7e35c0>,
 <matplotlib.axis.XTick at 0x12a9d7e3dd8>,
 <matplotlib.axis.XTick at 0x12a9d7ec2b0>,
 <matplotlib.axis.XTick at 0x12a9d7ec748>,
 <matplotlib.axis.XTick at 0x12a9d7ecbe0>,
 <matplotlib.axis.XTick at 0x12a9d7f50b8>,
 <matplotlib.axis.XTick at 0x12a9d7f5550>],
[Text(0, 0, 'age'),
 Text(1, 0, 'job'),
 Text(2, 0, 'marital'),
 Text(3, 0, 'education'),
 Text(4, 0, 'default'),
 Text(5, 0, 'balance'),
 Text(6, 0, 'housing'),
 Text(7, 0, 'loan'),
 Text(8, 0, 'contact'),
 Text(9, 0, 'day'),
 Text(10, 0, 'month'),
 Text(11, 0, 'duration'),
 Text(12, 0, 'campaign'),
 Text(13, 0, 'pdays'),
 Text(14, 0, 'previous'),
 Text(15, 0, 'poutcome'),
 Text(16, 0, 'y')])
```

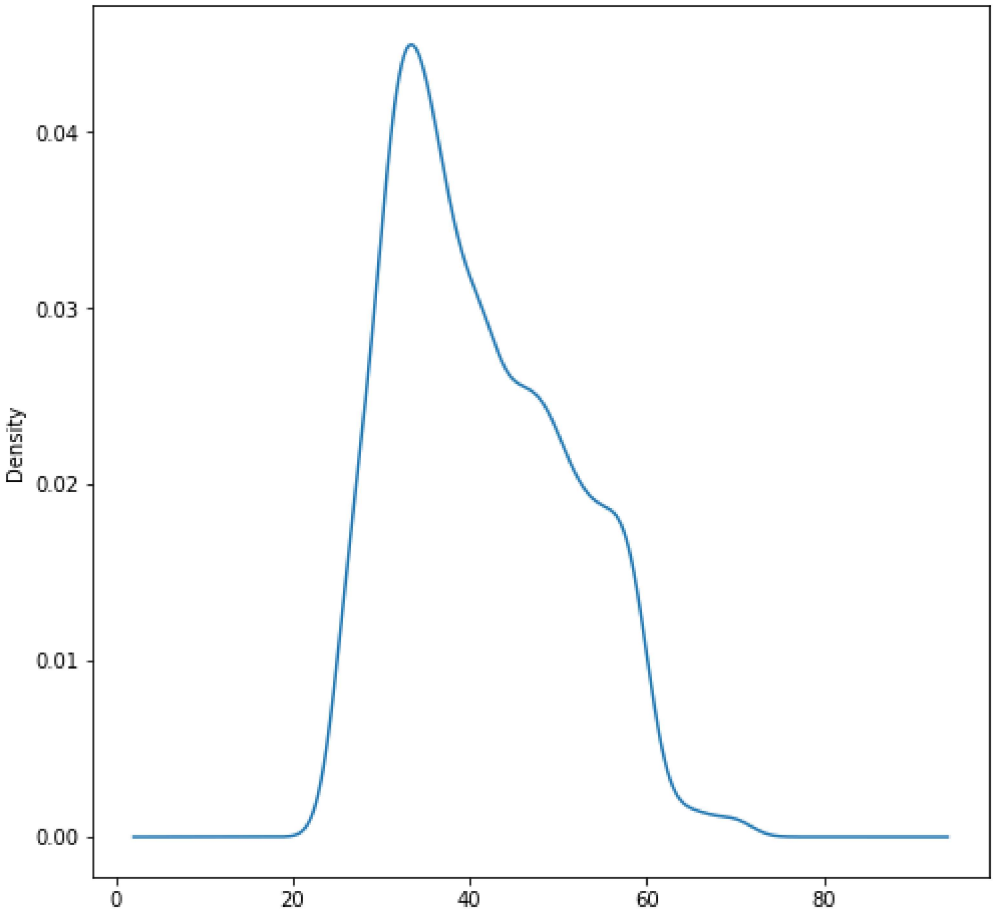


Outlier detection - Kernel Density Estimation

- Non-parametric way to estimate the probability density function of a given feature
- Can be advantageous compared to extreme value detection (e.g. Tukey IQR)
  - Capures outliers in bimodal distributions

In [77]:
```
new_df.age.plot.kde(figsize=(8,8))
```

Out[77]:
```
<AxesSubplot:ylabel='Density'>
```



In [80]:

```
from sklearn.preprocessing import scale
from statsmodels.nonparametric.kde import KDEUnivariate

def find_outliers_kde(x):
    x_scaled = scale(list(map(float, x)))
    kde = KDEUnivariate(x_scaled)
    kde.fit(bw="scott")
    pred = kde.evaluate(x_scaled)

    n = sum(pred < 0.01)
    outlier_ind = np.asarray(pred).argsort()[:n]
    outlier_value = np.asarray(x)[outlier_ind]

    return outlier_ind, outlier_value
```

In [81]:

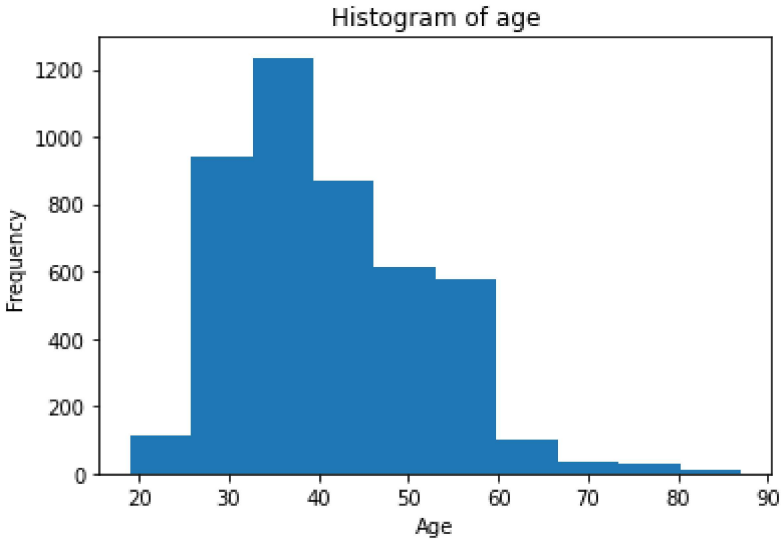
```
kde_indices, kde_values = find_outliers_kde(df['age'])
print(np.sort(kde_values))
```

```
[19 19 19 19 76 76 77 77 77 77 77 77 78 78 78 79 79 79 79 80 80 80 80 80
 80 81 83 83 83 83 84 86 87]
```

In [82]:

```
plt.hist(df.age.values)
plt.title("Histogram of age")
plt.xlabel("Age")
plt.ylabel("Frequency")
```

Out[82]: Text(0, 0.5, 'Frequency')

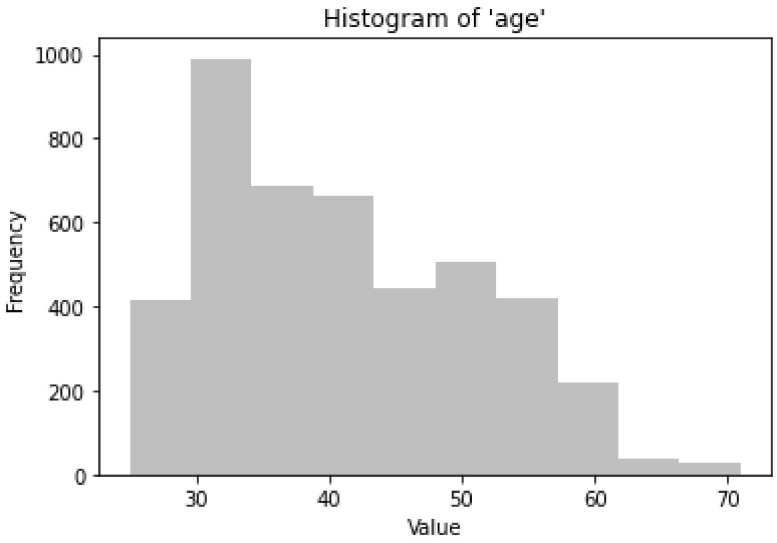


B. Distribution of Features

- A histogram is a simple representation of the distribution of values for a given feature
- X-axis represents value bins and y-axis represents the frequency of an observation falling into that bin
- It is also interesting to look at distributions broken up by outcome categories

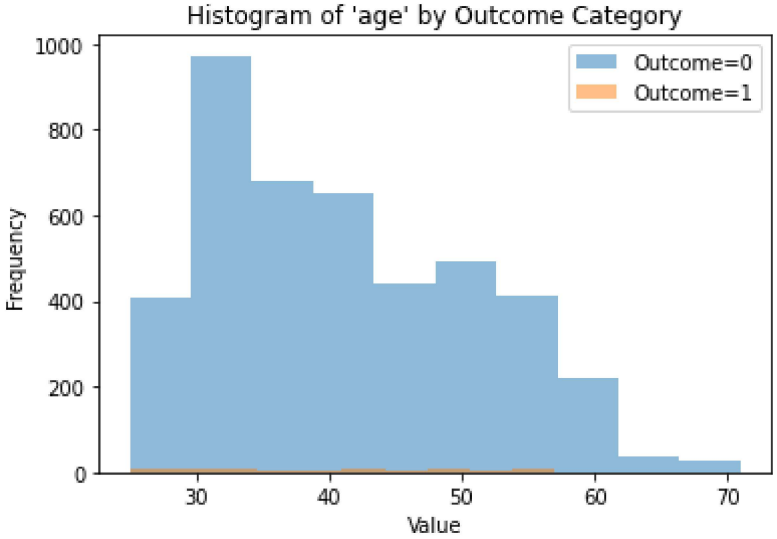
In [83]:
```
# Use pyplot in matplotlib to plot histograms
def plot_histogram(x):
    plt.hist(x, color='gray', alpha=0.5)
    plt.title("Histogram of '{var_name}'".format(var_name=x.name))
    plt.xlabel("Value")
    plt.ylabel("Frequency")
    plt.show()
```

In [84]:
```
plot_histogram(new_df['age'])
```



In [85]:
```
def plot_histogram_dv(x,y):
    plt.hist(list(x[y==0]), alpha=0.5, label='Outcome=0')
    plt.hist(list(x[y==1]), alpha=0.5, label='Outcome=1')
    plt.title("Histogram of '{var_name}' by Outcome
Category".format(var_name=x.name))
    plt.xlabel("Value")
    plt.ylabel("Frequency")
    plt.legend(loc='upper right')
    plt.show()
```

In [86]:
```
plot_histogram_dv(new_df['age'], y)
```



Part4: Feature Engineering

A. Interactions between features

B. Dimensionality reduction using PCA

A. Interactions amongst features

- A simple two-way interaction is represented by:
  - X3 = X1 * X2, where X3 is the interaction between X1 and X2
- Can add interaction terms as additional new features to your model; useful for model if the impact of two or more features on the outcome is non-additive
- Example:
  - Interaction: education and political ideology; outcome: concerns about climate change
  - While an increase in education amongst liberals or moderates increases concerns about climate change, an increase in education amongst conservatives has the opposite effect
  - The education-political ideology interaction captures more than the two features alone
- Although it is very easy to calculate two-way interactions amongst all features, it is very computationally expensive
  - 10 features = 45 two-way interaction terms
  - 50 features = 1,225 two-way interaction terms
  - 100 features = 4,950 two-way interaction terms
  - 500 features = 124,750 two-way interation terms
  - Recommend understanding your data and domain if possible and selectively choosing interaction terms

In [87]:
```
from IPython.display import Image
Image(filename=r'C:\Users\abhir\OneDrive\Desktop\bank\interactions.jpg')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-87-e487a6550b5f> in <module>
      1 from IPython.display import Image
----> 2 Image(filename=r'C:\Users\abhir\OneDrive\Desktop\bank\interactions.jpg')

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in __init__(self, dat
a, url, filename, format, embed, width, height, retina, unconfined, metadata)
   1223         self.unconfined = unconfined
   1224         super(Image, self).__init__(data=data, url=url, filename=filename,
-> 1225                 metadata=metadata)
   1226
   1227         if self.width is None and self.metadata.get('width', {}):

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in __init__(self, dat
a, url, filename, metadata)
    628             self.metadata = {}
    629
--> 630         self.reload()
    631         self._check_data()
    632

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in reload(self)
   1254         """Reload the raw data from file or URL."""
   1255         if self.embed:
-> 1256             super(Image,self).reload()
   1257             if self.retina:
```

```
   1258                 self._retina_shape()

~\Anaconda3\NM\envs\py36\lib\site-packages\IPython\core\display.py in reload(self)
    653         """Reload the raw data from file or URL."""
    654         if self.filename is not None:
--> 655             with open(self.filename, self._read_flags) as f:
    656                 self.data = f.read()
    657         elif self.url is not None:

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\abhir\\OneDrive\\Des
ktop\\bank\\interactions.jpg'
```

In [88]:

```
# Use PolynomialFeatures in sklearn.preprocessing to create two-way
interactions for all features
from itertools import combinations
from sklearn.preprocessing import PolynomialFeatures

def add_interactions(df):
    # Get feature names
    combos = list(combinations(list(df.columns), 2))
    colnames = list(df.columns) + ['_'.join(x) for x in combos]

    # Find interactions
    poly = PolynomialFeatures(interaction_only=True,include_bias=False)
    df = poly.fit_transform(df)
    df = pd.DataFrame(df)
    df.columns = colnames

    # Remove interaction terms with all 0 values
    noint_indicies = [i for i, x in enumerate(list((df == 0).all())) if x]
    df = df.drop(df.columns[noint_indicies], axis=1)

    return df
```

In [89]:

```
np.any(np.isnan(new_df))
```

Out[89]: True

In [90]:

```
new_df.head(20)
```

Out[90]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	c
0	30.0	0.0	1.0	0.0	0.0	0.068455	0.0	0.0	1.0	19.0	10.0	0.024826	
1	33.0	1.0	1.0	1.0	0.0	0.108750	1.0	1.0	1.0	11.0	5.0	0.071500	
2	35.0	2.0	0.0	2.0	0.0	0.062590	1.0	0.0	1.0	16.0	4.0	0.059914	

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	c
3	30.0	2.0	1.0	2.0	0.0	0.064281	1.0	1.0	1.0	3.0	6.0	0.064548	
4	59.0	3.0	1.0	1.0	0.0	0.044469	1.0	0.0	1.0	5.0	5.0	0.073486	
5	35.0	2.0	0.0	2.0	0.0	0.054496	0.0	0.0	1.0	23.0	2.0	0.045349	
6	36.0	4.0	1.0	2.0	0.0	0.048590	1.0	0.0	1.0	14.0	5.0	0.111552	
7	39.0	5.0	1.0	1.0	0.0	0.046442	1.0	0.0	1.0	6.0	5.0	0.048659	
8	41.0	6.0	1.0	2.0	0.0	0.047436	1.0	0.0	1.0	14.0	5.0	0.017544	
9	43.0	1.0	1.0	0.0	0.0	0.043288	1.0	1.0	1.0	17.0	4.0	0.102284	
10	39.0	1.0	1.0	1.0	0.0	0.170293	1.0	0.0	1.0	20.0	5.0	0.089043	
11	43.0	7.0	1.0	1.0	0.0	0.048013	1.0	0.0	1.0	17.0	4.0	0.036081	
12	36.0	5.0	1.0	2.0	0.0	0.059355	0.0	0.0	1.0	13.0	8.0	0.107249	
13	NaN	8.0	0.0	1.0	0.0	0.051207	0.0	0.0	1.0	30.0	4.0	0.085071	
14	31.0	3.0	1.0	1.0	0.0	0.049301	1.0	1.0	1.0	29.0	1.0	0.028136	
15	40.0	2.0	1.0	2.0	0.0	0.047073	0.0	1.0	1.0	29.0	8.0	0.061238	
16	56.0	5.0	1.0	1.0	0.0	0.099140	0.0	0.0	1.0	27.0	8.0	0.077789	
17	37.0	7.0	0.0	2.0	0.0	0.075569	1.0	0.0	1.0	20.0	4.0	0.036412	
18	25.0	3.0	0.0	0.0	0.0	0.041503	1.0	0.0	1.0	23.0	5.0	0.081430	
19	31.0	1.0	1.0	1.0	0.0	0.046241	0.0	0.0	1.0	7.0	7.0	0.047666	

In [91]:

```
new_df.dropna(inplace=True)
X = add_interactions(new_df)
print(X.head(20))
```

```
     age  job  marital  education  default   balance  housing  loan  contact  \
0   30.0  0.0      1.0        0.0      0.0  0.068455      0.0   0.0      1.0
1   33.0  1.0      1.0        1.0      0.0  0.108750      1.0   1.0      1.0
2   35.0  2.0      0.0        2.0      0.0  0.062590      1.0   0.0      1.0
3   30.0  2.0      1.0        2.0      0.0  0.064281      1.0   1.0      1.0
4   59.0  3.0      1.0        1.0      0.0  0.044469      1.0   0.0      1.0
5   35.0  2.0      0.0        2.0      0.0  0.054496      0.0   0.0      1.0
6   36.0  4.0      1.0        2.0      0.0  0.048590      1.0   0.0      1.0
7   39.0  5.0      1.0        1.0      0.0  0.046442      1.0   0.0      1.0
8   41.0  6.0      1.0        2.0      0.0  0.047436      1.0   0.0      1.0
9   43.0  1.0      1.0        0.0      0.0  0.043288      1.0   1.0      1.0
10  39.0  1.0      1.0        1.0      0.0  0.170293      1.0   0.0      1.0
11  43.0  7.0      1.0        1.0      0.0  0.048013      1.0   0.0      1.0
12  36.0  5.0      1.0        2.0      0.0  0.059355      0.0   0.0      1.0
13  31.0  3.0      1.0        1.0      0.0  0.049301      1.0   1.0      1.0
14  40.0  2.0      1.0        2.0      0.0  0.047073      0.0   1.0      1.0
15  56.0  5.0      1.0        1.0      0.0  0.099140      0.0   0.0      1.0
16  37.0  7.0      0.0        2.0      0.0  0.075569      1.0   0.0      1.0
17  25.0  3.0      0.0        0.0      0.0  0.041503      1.0   0.0      1.0
18  31.0  1.0      1.0        1.0      0.0  0.046241      0.0   0.0      1.0
19  38.0  2.0      1.0        1.0      0.0  0.044469      1.0   0.0      1.0

     day  ...  campaign_pdays  campaign_previous  campaign_poutcome  \
```

```
    0   19.0  ...        0.000000              0.0               0.0
    1   11.0  ...        0.389908              4.0               0.0
    2   16.0  ...        0.379587              1.0               0.0
    3    3.0  ...        0.000000              0.0               0.0
    4    5.0  ...        0.000000              0.0               0.0
    5   23.0  ...        0.405963              6.0               0.0
    6   14.0  ...        0.379587              2.0               1.0
    7    6.0  ...        0.000000              0.0               0.0
    8   14.0  ...        0.000000              0.0               0.0
    9   17.0  ...        0.169725              2.0               0.0
    10  20.0  ...        0.000000              0.0               0.0
    11  17.0  ...        0.000000              0.0               0.0
    12  13.0  ...        0.000000              0.0               0.0
    13  29.0  ...        0.277523              1.0               0.0
    14  29.0  ...        0.000000              0.0               0.0
    15  27.0  ...        0.000000              0.0               0.0
    16  20.0  ...        0.175459              2.0               0.0
    17  23.0  ...        0.000000              0.0               0.0
    18   7.0  ...        0.175459              1.0               1.0
    19  18.0  ...        0.000000              0.0               0.0

        campaign_y  pdays_previous  pdays_poutcome  pdays_y  previous_poutcome  \
    0          0.0        0.000000        0.000000      0.0                0.0
    1          0.0        1.559633        0.000000      0.0                0.0
    2          0.0        0.379587        0.000000      0.0                0.0
    3          0.0        0.000000        0.000000      0.0                0.0
    4          0.0        0.000000        0.000000      0.0                0.0
    5          0.0        0.608945        0.000000      0.0                0.0
    6          0.0        0.759174        0.379587      0.0                2.0
    7          0.0        0.000000        0.000000      0.0                0.0
    8          0.0        0.000000        0.000000      0.0                0.0
    9          0.0        0.339450        0.000000      0.0                0.0
    10         0.0        0.000000        0.000000      0.0                0.0
    11         0.0        0.000000        0.000000      0.0                0.0
    12         0.0        0.000000        0.000000      0.0                0.0
    13         0.0        0.277523        0.000000      0.0                0.0
    14         0.0        0.000000        0.000000      0.0                0.0
    15         0.0        0.000000        0.000000      0.0                0.0
    16         0.0        0.350917        0.000000      0.0                0.0
    17         0.0        0.000000        0.000000      0.0                0.0
    18         0.0        0.175459        0.175459      0.0                1.0
    19         0.0        0.000000        0.000000      0.0                0.0

        previous_y  poutcome_y
    0          0.0         0.0
    1          0.0         0.0
    2          0.0         0.0
    3          0.0         0.0
    4          0.0         0.0
    5          0.0         0.0
    6          0.0         0.0
    7          0.0         0.0
    8          0.0         0.0
    9          0.0         0.0
    10         0.0         0.0
    11         0.0         0.0
    12         0.0         0.0
    13         0.0         0.0
    14         0.0         0.0
    15         0.0         0.0
    16         0.0         0.0
    17         0.0         0.0
    18         0.0         0.0
    19         0.0         0.0
```

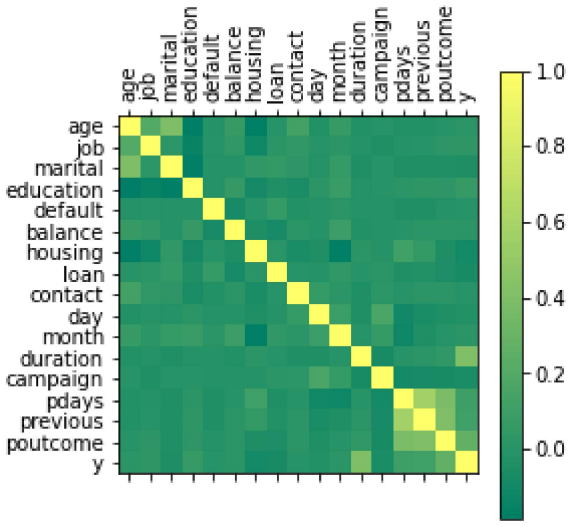
[20 rows x 153 columns]

In [92]:
```
new_df.corr()
```

Out[92]:

	age	job	marital	education	default	balance	housing	loan	cont
age	1.000000	0.204509	0.400202	-0.183500	-0.012006	0.076988	-0.175128	0.002801	0.138
job	0.204509	1.000000	0.024913	-0.143833	0.004901	0.040147	-0.106154	0.022017	0.042
marital	0.400202	0.024913	1.000000	-0.171556	-0.011295	-0.009263	0.042407	0.051428	0.032
education	-0.183500	-0.143833	-0.171556	1.000000	-0.015077	0.058158	-0.081211	-0.025618	-0.059
default	-0.012006	0.004901	-0.011295	-0.015077	1.000000	-0.070774	0.002233	0.066143	-0.018
balance	0.076988	0.040147	-0.009263	0.058158	-0.070774	1.000000	-0.045919	-0.070321	0.020
housing	-0.175128	-0.106154	0.042407	-0.081211	0.002233	-0.045919	1.000000	0.013277	-0.055
loan	0.002801	0.022017	0.051428	-0.025618	0.066143	-0.070321	0.013277	1.000000	0.002
contact	0.138571	0.042699	0.032980	-0.059884	-0.018757	0.020901	-0.055726	0.002748	1.000
day	-0.020278	0.002407	-0.008946	0.018935	-0.015504	-0.005183	-0.033938	-0.006607	0.062
month	0.073971	0.021852	0.063980	0.083242	0.010928	0.102215	-0.170038	0.040592	0.022
duration	-0.019929	-0.016276	-0.031614	-0.008091	-0.010037	-0.018864	0.019767	-0.002998	-0.026
campaign	-0.002761	-0.039266	0.008274	0.008227	-0.011479	-0.007751	-0.005764	0.018397	0.028
pdays	-0.020031	-0.004720	-0.023557	0.014035	-0.025440	0.005396	0.121777	-0.029094	0.008
previous	-0.020426	0.008944	-0.037952	0.034662	-0.025911	0.017672	0.048686	-0.019258	0.013
poutcome	0.005833	0.030090	-0.024152	0.032386	-0.015255	0.018973	-0.043531	-0.051381	0.018
y	0.016912	0.035704	-0.048986	0.067472	0.004623	0.015822	-0.094919	-0.066805	0.009

In [93]:
```
plt.matshow(new_df.corr(),cmap="summer")
plt.colorbar()
plt.xticks(list(range(len(new_df.columns))),new_df.columns,rotation="vertical

plt.yticks(list(range(len(new_df.columns))),new_df.columns,rotation="horizont



plt.show()
```



In [94]:
```
new_df.corr()["y"].sort_values(ascending=False)
```

Out[94]:
```
y            1.000000
duration     0.404098
poutcome     0.277297
previous     0.113605
pdays        0.104959
education    0.067472
job          0.035704
month        0.019484
age          0.016912
balance      0.015822
contact      0.009177
default      0.004623
day         -0.017893
marital     -0.048986
campaign    -0.056475
loan        -0.066805
housing     -0.094919
Name: y, dtype: float64
```

In [95]:
```
new_df.to_csv(r"C:\Users\abhir\OneDrive\Desktop\bank\preprocessed.csv",index=
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-95-5346f1b197fd> in <module>
----> 1 new_df.to_csv(r"C:\Users\abhir\OneDrive\Desktop\bank\preprocessed.csv",index=Fa
lse)

~\AppData\Roaming\Python\Python36\site-packages\pandas\core\generic.py in to_csv(self, p
ath_or_buf, sep, na_rep, float_format, columns, header, index, index_label, mode, encodi
ng, compression, quoting, quotechar, line_terminator, chunksize, date_format, doublequot
e, escapechar, decimal, errors)
   3168             decimal=decimal,
   3169         )
-> 3170         formatter.save()
   3171
   3172         if path_or_buf is None:

~\AppData\Roaming\Python\Python36\site-packages\pandas\io\formats\csvs.py in save(self)
    188                 encoding=self.encoding,
```

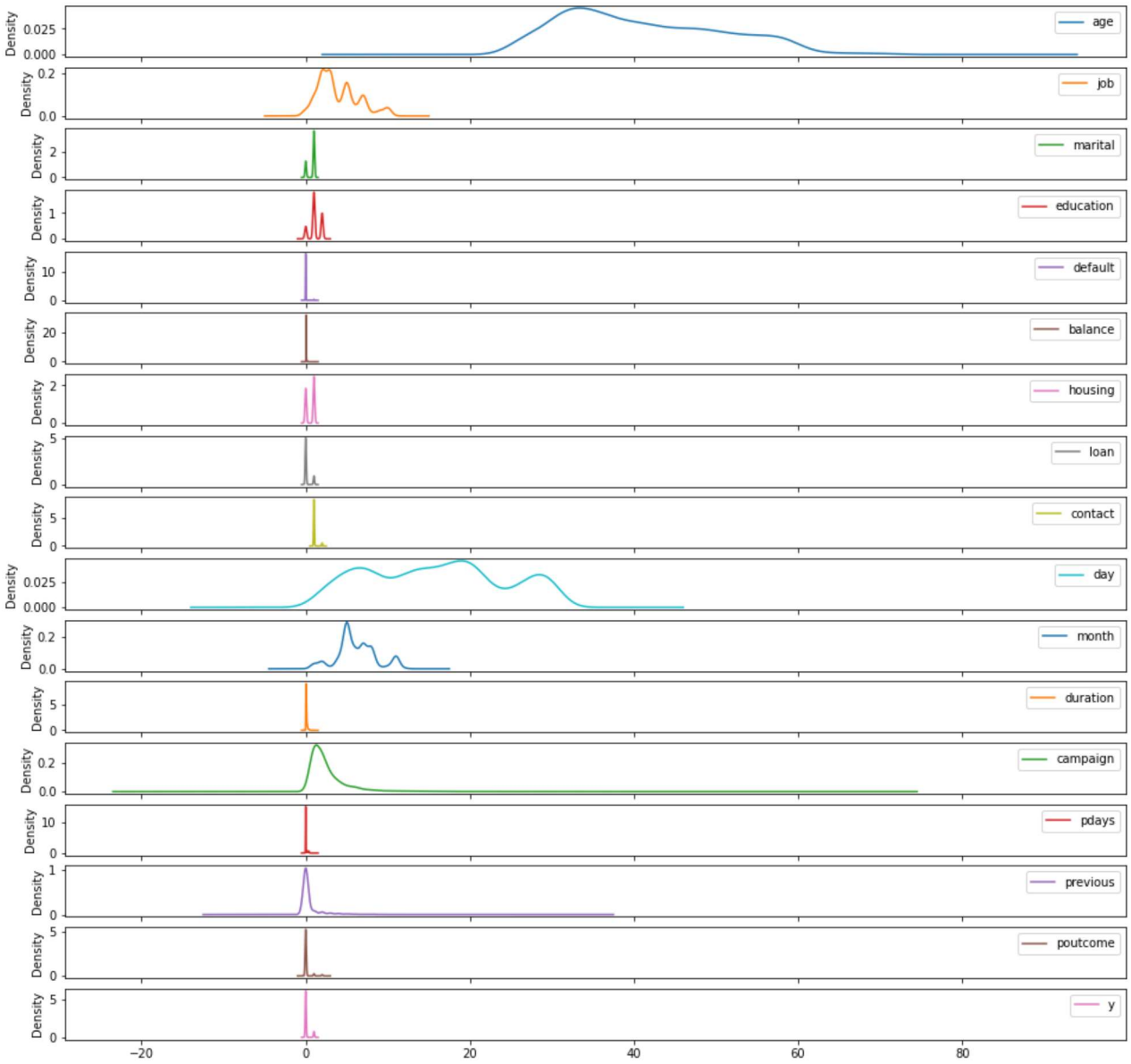
```
    189                 errors=self.errors,
--> 190                 compression=dict(self.compression_args, method=self.compression
),
    191             )
    192             close = True

~\AppData\Roaming\Python\Python36\site-packages\pandas\io\common.py in get_handle(path_o
r_buf, mode, encoding, compression, memory_map, is_text, errors)
    491         if encoding:
    492             # Encoding
--> 493             f = open(path_or_buf, mode, encoding=encoding, errors=errors, newli
ne="")
    494         elif is_text:
    495             # No explicit encoding

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\abhir\\OneDrive\\Des
ktop\\bank\\preprocessed.csv'
```

In [98]:
```
new_df.y.unique()
```

Out[98]:
```
array([0., 1.])
```

In [99]:
```
new_df.plot.kde(subplots=True,figsize=(16,16))
```

Out[99]:
```
array([<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>], dtype=object)
```

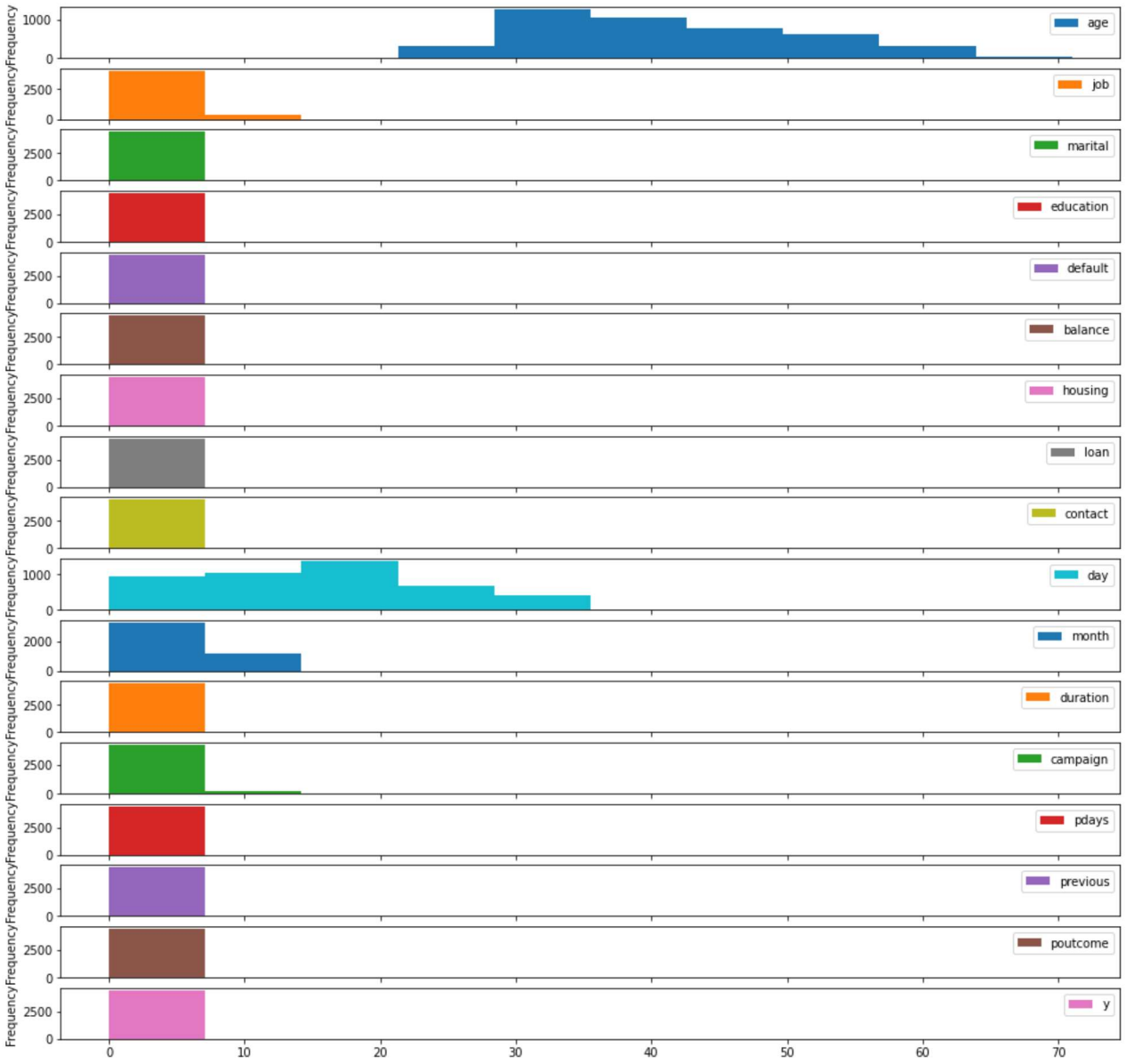


In [100…

```
new_df.plot.hist(subplots=True,figsize=(16,16))
```

Out[100…

```
array([<AxesSubplot:ylabel='Frequency'>, <AxesSubplot:ylabel='Frequency'>,
       <AxesSubplot:ylabel='Frequency'>, <AxesSubplot:ylabel='Frequency'>,
       <AxesSubplot:ylabel='Frequency'>, <AxesSubplot:ylabel='Frequency'>,
       <AxesSubplot:ylabel='Frequency'>, <AxesSubplot:ylabel='Frequency'>,
       <AxesSubplot:ylabel='Frequency'>, <AxesSubplot:ylabel='Frequency'>,
       <AxesSubplot:ylabel='Frequency'>], dtype=object)
```


Frequency
age
job
marital
education
default
balance
housing
loan
contact
day
month
duration
campaign
pdays
previous
poutcome
y
0
10
20
30
40
50
60
70

In [ ]:

In [ ]:

In [ ]:

In [ ]: