Assignment -5

LDA & PCA

2148059

In [35]:
```python
import pandas as pd
```

In [36]:
```python
df=pd.read_csv("/Users/persie/Downloads/bank-12.csv")
```

In [37]:
```python
df.head(5)
```

Out[37]:

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

In [38]:
```python
#value counts of the target variable
df["y"].value_counts()
```

Out[38]:
```
no      4000
yes      521
Name: y, dtype: int64
```

In [39]:
```python
rem=["contact","day"]
df=df.drop(rem,axis=1)
df.columns
```

Out[39]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance',
'housing',
       'loan', 'month', 'duration', 'campaign', 'pdays', 'previous
',
       'poutcome', 'y'],
      dtype='object')
```

In [40]:
```python
#assigning 1 if target variable is yes and 0 if target is no
df["y"]=[1 if x=="yes" else 0 for x in df["y"]]

#x as dataframe of features and y as the target variable
x=df.drop("y",1)
y=df.y
```

In [41]:
```python
x.head(5)
```

Out[41]:

	age	job	marital	education	default	balance	housing	loan	month	duration
0	30	unemployed	married	primary	no	1787	no	no	oct	79
1	33	services	married	secondary	no	4789	yes	yes	may	220
2	35	management	single	tertiary	no	1350	yes	no	apr	185
3	30	management	married	tertiary	no	1476	yes	yes	jun	199
4	59	blue-collar	married	secondary	no	0	yes	no	may	226

In [42]:
```python
y.head(5)
```

Out[42]:
```
0    0
1    0
2    0
3    0
4    0
Name: y, dtype: int64
```

Data Cleaning

A. dealing with the data types

converting categorical data to numerical data

In [43]:
```python
#categorical variable
x["marital"].head()
```

Out[43]:
```
0    married
1    married
2     single
3    married
4    married
Name: marital, dtype: object
```

In [44]:
```python
#cheking the no of categories in all the features
for col_names in x.columns:
    if x[col_names].dtype=="object":
        cat=len(x[col_names].unique())
        print("features: {col_names} has {cat} categories".format(c
```

```
features: job has 12 categories
features: marital has 3 categories
features: education has 4 categories
features: default has 2 categories
features: housing has 2 categories
features: loan has 2 categories
features: month has 12 categories
features: poutcome has 4 categories
```

Categorise all the other features exceopth month and job

In [45]:
```
#list of features to dummy
todummy=["marital","education","default","housing","loan","poutcome
```

In [46]:
```
#function to dummy all the categorical variables for modelling
def dummy(df,todummy):
    for x in todummy:
        dummies=pd.get_dummies(df[x],prefix=x,dummy_na=False)
        df=df.drop(x,1)
        df=pd.concat([df,dummies],axis=1)
    return df
```

In [47]:
```
x= dummy(x,todummy)
x.head(5)
```

Out[47]:

	age	balance	duration	campaign	pdays	previous	marital_divorced	marital_married	n
0	30	1787	79	1	-1	0	0	1	
1	33	4789	220	1	339	4	0	1	
2	35	1350	185	1	330	1	0	0	
3	30	1476	199	4	-1	0	0	1	
4	59	0	226	1	-1	0	0	1	

5 rows × 47 columns

In [48]:
```
x.columns
```

Out[48]:
```
Index(['age', 'balance', 'duration', 'campaign', 'pdays', 'previou
s',
       'marital_divorced', 'marital_married', 'marital_single',
       'education_primary', 'education_secondary', 'education_tert
iary',
       'education_unknown', 'default_no', 'default_yes', 'housing_
no',
       'housing_yes', 'loan_no', 'loan_yes', 'poutcome_failure',
       'poutcome_other', 'poutcome_success', 'poutcome_unknown', '
job_admin.',
       'job_blue-collar', 'job_entrepreneur', 'job_housemaid',
       'job_management', 'job_retired', 'job_self-employed', 'job_
services',
       'job_student', 'job_technician', 'job_unemployed', 'job_unk
nown',
       'month_apr', 'month_aug', 'month_dec', 'month_feb', 'month_
jan',
       'month_jul', 'month_jun', 'month_mar', 'month_may', 'month_
nov',
       'month_oct', 'month_sep'],
      dtype='object')
```

b. handling missing values

In [49]:
```
x.isnull().sum().sort_values(ascending=True)
```

Out[49]:
```
age                    0
job_entrepreneur       0
job_housemaid          0
job_management         0
job_retired            0
job_self-employed      0
job_services           0
job_student            0
job_technician         0
job_unemployed         0
job_blue-collar        0
job_unknown            0
month_aug              0
month_dec              0
month_feb              0
month_jan              0
month_jul              0
month_jun              0
month_mar              0
month_may              0
month_nov              0
month_apr              0
month_oct              0
job_admin.             0
poutcome_success       0
balance                0
duration               0
campaign               0
pdays                  0
previous               0
marital_divorced       0
marital_married        0
marital_single         0
education_primary      0
poutcome_unknown       0
education_secondary    0
education_unknown      0
default_no             0
default_yes            0
housing_no             0
housing_yes            0
loan_no                0
loan_yes               0
poutcome_failure       0
poutcome_other         0
education_tertiary     0
month_sep              0
dtype: int64
```

there is no missing values in the data

the above values are the outliers

In [50]:
```
x.head(5)
```

Out[50]:

	age	balance	duration	campaign	pdays	previous	marital_divorced	marital_married	n
0	30	1787	79	1	-1	0	0	1	
1	33	4789	220	1	339	4	0	1	
2	35	1350	185	1	330	1	0	0	
3	30	1476	199	4	-1	0	0	1	
4	59	0	226	1	-1	0	0	1	

5 rows × 47 columns

PCA

In [51]:
```
from sklearn.preprocessing import StandardScaler
```

In [52]:
```
df=pd.DataFrame(x)
```

In [53]:
```
y_target=pd.DataFrame(y)
```

In [54]:
```
scalar=StandardScaler()
```

In [55]:
```
scalar.fit(df)
```

Out[55]:
```
StandardScaler()
```

In [56]:
```
scalar_data=scalar.transform(df)
```

In [57]:
```
#importing PCA
from sklearn.decomposition import PCA
```

In [104]:

```
#components=2
pca=PCA(n_components=10)
pca.fit(scalar_data)
x_pca=pca.transform(scalar_data)

print(x_pca.shape)
pca_df=pd.DataFrame(x_pca)
pca_df.head()
```

```
(4521, 10)
```

Out[104]:

	0	1	2	3	4	5	6	7	
0	-0.881322	1.654261	1.640155	-0.408573	-0.837181	0.671860	1.467470	0.254911	(
1	4.664252	-2.135589	0.953058	2.355156	0.813990	-2.101071	-0.653068	-0.243735	(
2	4.491902	1.489892	-2.184596	-0.710582	1.610306	-0.343370	0.375299	0.362673	-(
3	-1.107909	-0.227722	-0.692657	0.510633	3.334351	-2.305847	0.216432	-0.930444	(
4	-0.175611	-2.285620	1.393156	-1.397413	-0.201409	0.520988	-0.452758	-0.096852	-(

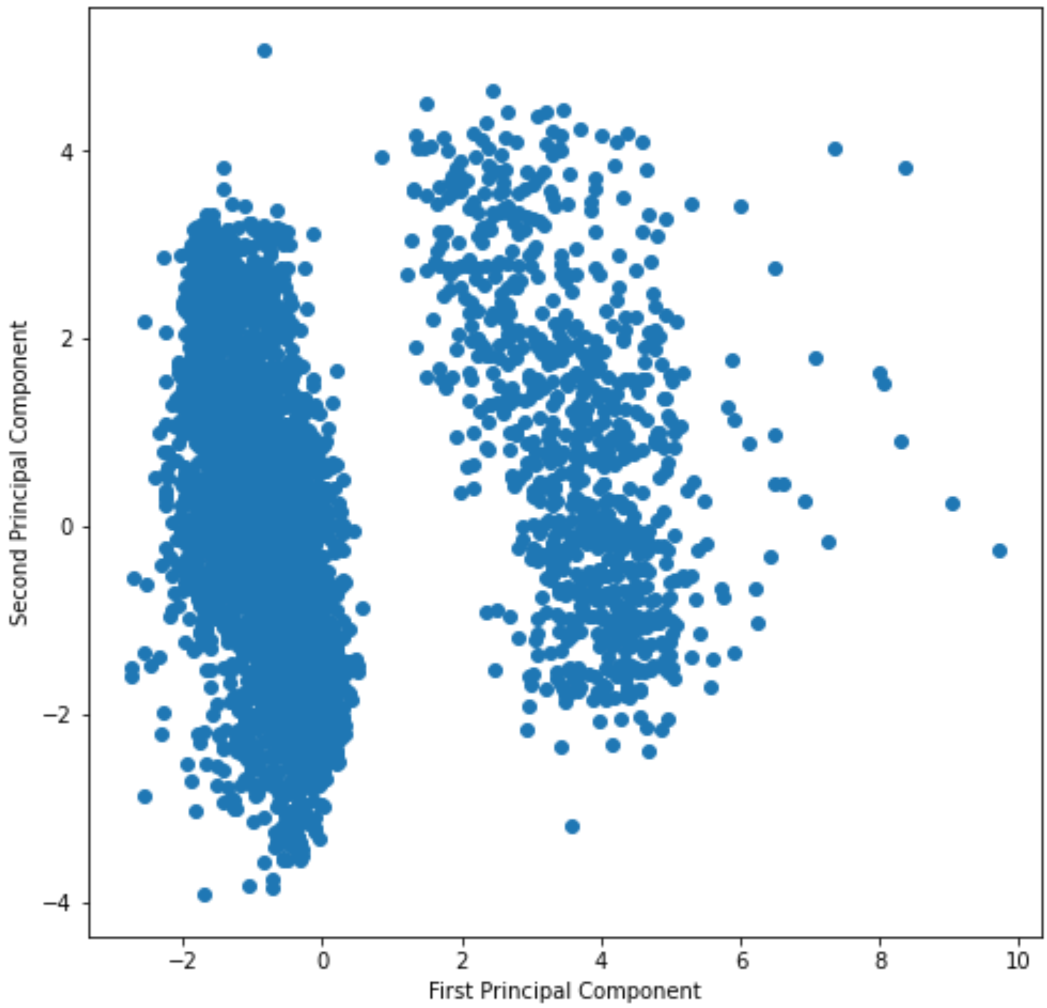
In [105]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#giving a larger plot
plt.figure(figsize=(8,8))

#plt.scatter(x_pca[:,0],x_pca[:,1],c=y_train,cmap='plasma')
plt.scatter(x_pca[:,0],x_pca[:,1])

#labelling x and y axes
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
```

Out[105]:
```
Text(0, 0.5, 'Second Principal Component')
```



LDA

In [92]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.5)
```

In [93]:
```
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [94]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysi
lda=LDA(n_components=1)
x_train=lda.fit_transform(x_train,y_train)
x_test=lda.transform(x_test)
lda_df=pd.DataFrame(x_test)
lda_df.head(5)
```

Out[94]:

	0
0	-1.181641
1	-0.451614
2	2.114893
3	0.450902
4	1.688223

In [101]:
```
from sklearn.ensemble import RandomForestClassifier

classifier=RandomForestClassifier(max_depth=2,random_state=0)

classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
y_pred2=classifier.predict(x_train)

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


print('Accuracy'+str(accuracy_score(y_train,y_pred2)))
```

```
Accuracy0.9061946902654867
```

In [103]:
```
cm=confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1956   38]
 [ 198   69]]
```

In [ ]:
```
```