EDA 04- Featuure Selection -I

1. Necessary Imports

In [25]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
import scipy.stats as s
import seaborn as sns
```

2. Reading dataset into CSV & Basic Data Description

a) Reading Data

In [87]:
```
data=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 4/Finance.csv")
```

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

Out[4]:

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

In [5]:
```
data.shape
```

Out[5]:
```
(45211, 17)
```

b) Check the Data Types

In [79]:
```
data.dtypes
```

Out[79]:
```
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
```

3. Check Missing Data or Null Values

In [109]:
```
data.isnull().sum()
```

Out[109]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

4. Check for Data Quality Issues

a) For categorical attributes you can ispect the unique values

You can display all the unique values and based on domain kbowledge can decide if incrrect data exists or not.

In [10]:
```
data['job'].unique()
```

Out[10]:
```
array(['management', 'technician', 'entrepreneur', 'blue-collar',
       'unknown', 'retired', 'admin.', 'services', 'self-employed
',
       'unemployed', 'housemaid', 'student'], dtype=object)
```

If you do not have the doamin knowledge , then value counts may give you an idea about the possible incorrect values

In [11]:
```
data['job'].value_counts()
```

Out[11]:
```
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: job, dtype: int64
```

You can also use describe() to get a bit more information about the feature under consideration

In [12]:
```
data['job'].describe()
```

Out[12]:
```
count           45211
unique             12
top       blue-collar
freq             9732
Name: job, dtype: object
```

b) For numeric attributes you can inspect all the posible set of values

If you have a domain knowledge , the minimum and maximum values can spot if incorrect data is present or not

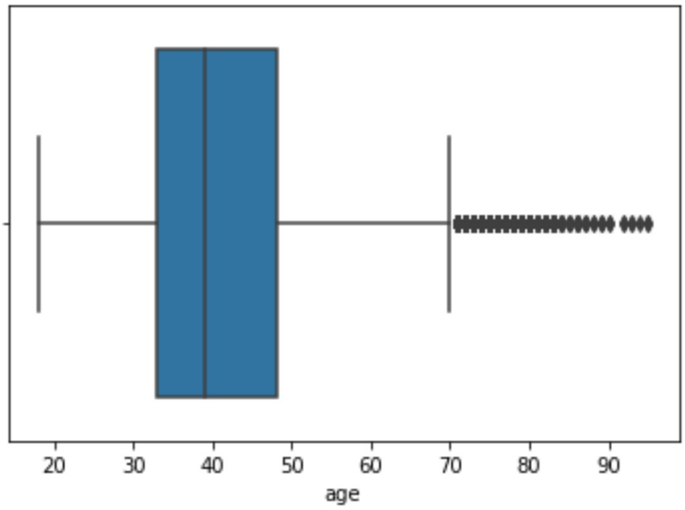
In [14]:
```
data['age'].describe()
```

Out[14]:
```
count    45211.000000
mean        40.936210
std         10.618762
min         18.000000
25%         33.000000
50%         39.000000
75%         48.000000
max         95.000000
Name: age, dtype: float64
```

If you do nat have the domain knowledge , the boxplot may give you some clue about the presence of possibly incorrect values. We call them as outliers. Outliers may be real or due to data collection problems.

In [19]:
```
sns.boxplot(x=data['age'], data=data)
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8e9b634a8>



5. Feature Selection Based on Filter Methods

a) Using Chi Square to test association between categorical attributes

The class variable is Y which is object type. So we can use Chi Square to check the association between all other object types and the class.The variables having strong association can be chosen as the features for machine learning algorithm.

Chi-Square Hypothesis : HO: There is no relationship / association / dependence between two attributes H1: There is a relationship / associatio/ dpendence between two variables

Lets perform a chi-square test of independence for 'default' and class variable 'y'

In [20]:
```
data['default'].value_counts()
```

Out[20]:
```
no     44396
yes      815
Name: default, dtype: int64
```

In [21]:
```
data['y'].value_counts()
```

Out[21]:
```
no     39922
yes     5289
Name: y, dtype: int64
```

In [22]:
```
ov=pd.crosstab(data['default'],data['y'])
```

In [23]:
```
ov
```

Out[23]:

y	**no**	**yes**
default		
no	39159	5237
yes	763	52

In [26]:
```
b=s.chi2_contingency(ov)
```

In [27]:
```
b
```

Out[27]:
```
(22.20224995571685,
 2.4538606753508344e-06,
 1,
 array([[39202.34261574,  5193.65738426],
        [  719.65738426,    95.34261574]]))
```

Here b is a tuple containing an immutable sequence of python objects . Here it contains four objcets . b[0] contains the value of chi2 statistic , b[1] contains the p-value of the test , b[2] contains the degree of freedom and b[3] contains the expected frequencies.

In [119]:
```
b[0]
```

Out[119]:
```
22.20224995571685
```

In [120]:
```
b[1]
```

Out[120]:
```
2.4538606753508344e-06
```

In [121]:
```
b[2]
```

Out[121]:
```
1
```

In [122]:
```
b[3]
```

Out[122]:
```
array([[39202.34261574,  5193.65738426],
       [  719.65738426,    95.34261574]])
```

Lets create a custom function to peform chi-square test of independence

In [123]:
```
def test_dependency(data,f1,f2,alpha):
    ov=pd.crosstab(data[f1],data[f2])
    b=s.chi2_contingency(ov)
    chi2_statistic=b[0]
    p_value=b[1]
    dof=b[2]
    critical_value=s.chi2.ppf(q=1-alpha, df=dof)
    print('Significance level: ',alpha)
    print('Degree of Freedom: ',dof)
    print('chi-square statistic:',chi2_statistic)
    print('critical_value:',critical_value)
    print('p-value:',p_value)

    if chi2_statistic>=critical_value:
        print("Reject H0,There is a relationship between 2 categori
cal variables")
    else:
        print("Retain H0,There is no relationship between 2 categor
ical variables")

    if p_value<=alpha:
        print("Reject H0,There is a relationship between 2 categori
cal variables")
    else:
        print("Retain H0,There is no relationship between 2 categor
ical variables")
```

In [126]:
```
test_dependency(data,'default','y',0.05)
```

```
Significance level:  0.05
Degree of Freedom:  1
chi-square statistic: 22.20224995571685
critical_value: 3.841458820694124
p-value: 2.4538606753508344e-06
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

In [125]:
```
test_dependency(data,'education','y',0.05)
```

```
Significance level:  0.05
Degree of Freedom:  3
chi-square statistic: 238.92350616407606
critical_value: 7.814727903251179
p-value: 1.6266562124072994e-51
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

Selecting k-Best Features based on Chi-Square Test

We will be using SelectKBest( )which takes numeric data only . So for that we have to encode all the categorical.We will be using mannual encoding for ordinal features whereas label encoding for all other nominal features.

Encoding Ordinal Features

In [59]:
```
ordinal_list=['education']
data['education'] = data['education'].replace(['primary','secondary
','tertiary','unknown'],[1,2,3,2])
```

In [60]:
```
data.head()
```

Out[60]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	du
0	58	4	1	3	0	2143	1	0	2	5	8	
1	44	9	2	2	0	29	1	0	2	5	8	
2	33	2	1	2	0	2	1	1	2	5	8	
3	47	1	1	2	0	1506	1	0	2	5	8	
4	33	11	2	2	0	1	0	0	2	5	8	

Encoding Nominal Features

In [56]:
```
nominal_list = []
for i in data.columns.tolist():
    if (data[i].dtype=='object') and (i not in ordinal_list):
        nominal_list.append(i)
print (nominal_list)
print('Number of nominal features:', str(len(nominal_list)))
```

```
['job', 'marital', 'default', 'housing', 'loan', 'contact', 'month
', 'poutcome', 'y']
Number of nominal features: 9
```

In [57]:
```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
for column in nominal_list:
    data[column]=encoder.fit_transform(data[column])
```

In [61]:
```
data.head()
```

Out[61]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	du
0	58	4	1	3	0	2143	1	0	2	5	8	
1	44	9	2	2	0	29	1	0	2	5	8	
2	33	2	1	2	0	2	1	1	2	5	8	
3	47	1	1	2	0	1506	1	0	2	5	8	
4	33	11	2	2	0	1	0	0	2	5	8	

Combining ordinal and nominal features after encoding

In [73]:
```
ordinal_data=data[ordinal_list]
nominal_data=data[nominal_list]
categorical_data = pd.concat([ordinal_data,nominal_data], axis=1)
```

In [72]:
```
categorical_data
```

Out[72]:

	job	marital	default	housing	loan	contact	month	poutcome	y
0	4	1	0	1	0	2	8	3	0
1	9	2	0	1	0	2	8	3	0
2	2	1	0	1	1	2	8	3	0
3	1	1	0	1	0	2	8	3	0
4	11	2	0	0	0	2	8	3	0
...	...	...	...	...	...	...	...	...	...
45206	9	1	0	0	0	0	9	3	1
45207	5	0	0	0	0	0	9	3	1
45208	5	1	0	0	0	0	9	2	1
45209	1	1	0	0	0	1	9	3	0
45210	2	1	0	0	0	0	9	1	0

45211 rows × 9 columns

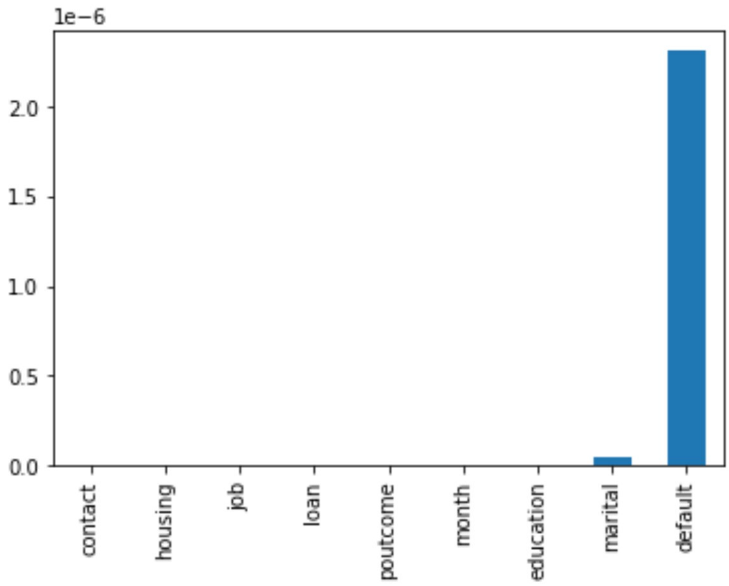
Selecting K Best Features based on Chi-Square Test

In [74]:
```
X=categorical_data.drop('y',axis=1)
Y=categorical_data['y']
chi_scores = chi2(X,Y)
```

In [75]:
```
p_values = pd.Series(chi_scores[1],index = X.columns)
p_values.sort_values(ascending = True , inplace = True)
chi2_values=pd.Series(chi_scores[0],index = X.columns)
chi2_values.sort_values(ascending = False , inplace = True)
```

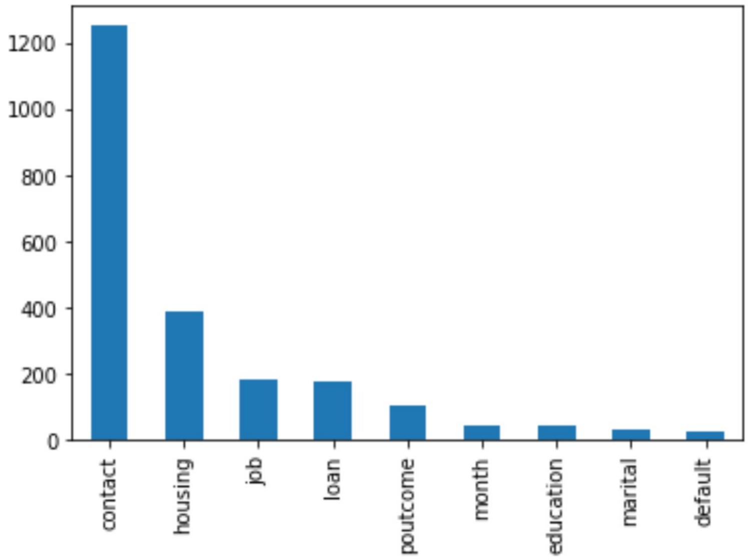
In [76]:
```
p_values.plot.bar()
```

Out[76]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8ea816748>



In [77]:
```
chi2_values.plot.bar()
```

Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8eb4231d0>



In [78]:
```
# Three features with highest chi-squared statistics are selected
chi2_features = SelectKBest(chi2, k = 3)
X_kbest_features = chi2_features.fit_transform(X, Y)

# Reduced features
print('Original feature number:', X.shape[1])
print('Reduced feature number:', X_kbest_features.shape[1])
```

```
Original feature number: 9
Reduced feature number: 3
```

In [81]:
```
index = chi2_features.get_support(indices=True)
print(index)
```

```
[1 4 6]
```

In [82]:
```
X
```

Out[82]:

	education	job	marital	default	housing	loan	contact	month	poutcome
0	3	4	1	0	1	0	2	8	3
1	2	9	2	0	1	0	2	8	3
2	2	2	1	0	1	1	2	8	3
3	2	1	1	0	1	0	2	8	3
4	2	11	2	0	0	0	2	8	3
...	...	...	...	...	...	...	...	...	...
45206	3	9	1	0	0	0	0	9	3
45207	1	5	0	0	0	0	0	9	3
45208	2	5	1	0	0	0	0	9	2
45209	2	1	1	0	0	0	1	9	3
45210	2	2	1	0	0	0	0	9	1

45211 rows × 9 columns

Features at index 1 , 4 and 6 are job , housing and contact respectively

In [85]:
```
selected_features= ['job','housing','contact']
```

b) Using Pearson Correlation Coefficient for Numeric Features vs Numeric Class

In [136]:
```
data=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 4/Finance.csv")
```

In [137]:
```
df=data.copy()
```

In [138]:
```
data['y'].dtype
```

Out[138]:
```
dtype('O')
```

In [139]:
```
data['y'].value_counts()
```

Out[139]:
```
no     39922
yes     5289
Name: y, dtype: int64
```

we can convert yes and no into 1 and 0 and change the data type from object to integer

In [140]:
```
data['y']=data['y'].apply(lambda x:0 if x=='no' else 1)
```

In [141]:
```
data['y'].value_counts()
```

Out[141]:
```
0    39922
1     5289
Name: y, dtype: int64
```

In [142]:
```
data['y'].dtype
```

Out[142]:
```
dtype('int64')
```

In [143]:
```
cor_matrix=data.corr()
```

In [144]:
```
print(cor_matrix['y'].sort_values(ascending=False))
```

```
y           1.000000
duration    0.394521
pdays       0.103621
previous    0.093236
balance     0.052838
age         0.025155
day        -0.028348
campaign   -0.073172
Name: y, dtype: float64
```

In [145]:
```
cor_matrix
```

Out[145]:

	age	balance	day	duration	campaign	pdays	previous	
age	1.000000	0.097783	-0.009120	-0.004648	0.004760	-0.023758	0.001288	0.02
balance	0.097783	1.000000	0.004503	0.021560	-0.014578	0.003435	0.016674	0.05
day	-0.009120	0.004503	1.000000	-0.030206	0.162490	-0.093044	-0.051710	-0.02
duration	-0.004648	0.021560	-0.030206	1.000000	-0.084570	-0.001565	0.001203	0.39
campaign	0.004760	-0.014578	0.162490	-0.084570	1.000000	-0.088628	-0.032855	-0.07
pdays	-0.023758	0.003435	-0.093044	-0.001565	-0.088628	1.000000	0.454820	0.10
previous	0.001288	0.016674	-0.051710	0.001203	-0.032855	0.454820	1.000000	0.09
y	0.025155	0.052838	-0.028348	0.394521	-0.073172	0.103621	0.093236	1.00

In [146]:
```
features_cor=(cor_matrix['y'].sort_values(ascending=False))
```

In [147]:
```
selected_num_features= ['duration','pdays','previous']
```

In [148]:
```
best_features=selected_features+(selected_num_features)
```

In [149]:
```
best_features
```

Out[149]:
```
['job', 'housing', 'contact', 'duration', 'pdays', 'previous']
```

In [151]:
```
data2=pd.concat([data[best_features],data['y']],axis=1)
```

In [152]:
```
data2
```

Out[152]:

	job	housing	contact	duration	pdays	previous	y
0	management	yes	unknown	261	-1	0	0
1	technician	yes	unknown	151	-1	0	0
2	entrepreneur	yes	unknown	76	-1	0	0
3	blue-collar	yes	unknown	92	-1	0	0
4	unknown	no	unknown	198	-1	0	0
...	...	...	...	...	...	...	...
45206	technician	no	cellular	977	-1	0	1
45207	retired	no	cellular	456	-1	0	1
45208	retired	no	cellular	1127	184	3	1
45209	blue-collar	no	telephone	508	-1	0	0
45210	entrepreneur	no	cellular	361	188	11	0

45211 rows × 7 columns

Remarks: Feature Selection using filter methods is independant of the machine learning model we use. Whether our seletced features will work better or not depends on the performance of ML model we apply.