INTERVIEW PREDICTION DOCUMENTATION

PREDICTION OF INTERVIEW ATTENDANCE USING MACHINE LEARNING AND PYTHON

ANIKET CHATTOPADHYA, RCCIIT, REG NO. 161170110007

CAMELIA MAHATO, RCCIIT, REG NO.161170110025

CHANDRABALI BISHNU, RCCIIT, REG NO.:161170110026

PALLAB CHAKRABORTY, RCCIIT, REG NO.:161170110048

SOHAM MANDAL, RCCIIT, REG NO.161170110070

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my project.

PROJECT OBJECTIVE

Our project objective was to analyze a data sheet on Interview prediction stats consisting around twenty eight data entries. We analyzed the data using various parameters and tried to cover almost every aspect of the given data and using Machine Learning and python programming. We represented our analysis in both statistically and pictorially. Finally we applied various models on the data, selected the best feature and calculated precision of the data.

DATA DESCRIPTION

Data Info:

df.info()

df.describe()

df.index()

df.columns()

Range Index : 1234 entries

Data columns : 28 columns

Column	Renamed
Date of interview	doi
Client name	cl_nam
Industry	indus
Location	cl_loc
Position to be closed	pos

Nature of Skillset	skill
Interview Type	intrvw_typ
Name(Cand ID)	cand_nam
Gender	gend
Candidate Current Location	cand_cur_loc
Candidate Job Location	cand_j_loc
Interview Venue	intrvw_ven
Candidate Native Location	cand_nat_loc
Have you obtained the necessary permission to start at the required time	enq_perm

Hope there will be no unscheduled meetings	enq_unsch_meet
Can I call you three hours before the interview and follow up on your attendance for the interview	enq_call
Can I have an alternative number/desk number. I assure you that I will not trouble you too much	enq_num
Have you taken a printout of your updated resume. Have you read the JD and understood the same	enq_resume
Are you clear with the venue details and the landmark	enq_ven

Has the call letter been shared	enq_call_letter
Expected Attendance	expc_at
Observed Attendance	obs_at
Marital Status	married

DATA LOADING

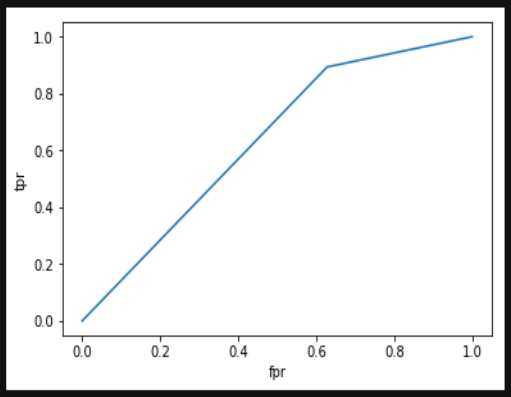
In order to access the data file in the python interpreter we first created a data frame in python using the provided data sheet and loaded the data in the interpreter using the data frame.

The python code for this process is given below:

df=pd.read_csv(“Interview.csv”)

DISTRIBUTION ANALYSIS

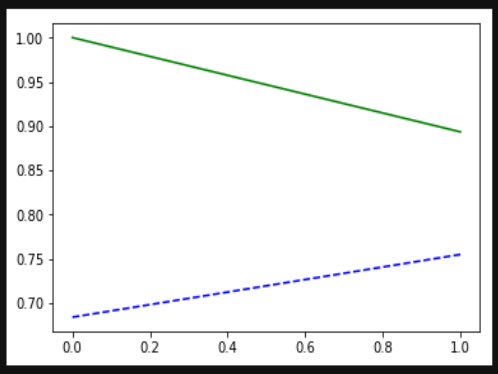
[233]: Text(0, 0.5, 'tpr')



[234]:
```
prec,recall,threshhold=metrics.precision_recall_curve(ytest,predict_test)
plt.plot(threshhold,prec[:-1],'b--')
plt.plot(threshhold,recall[:-1],'g-')
```

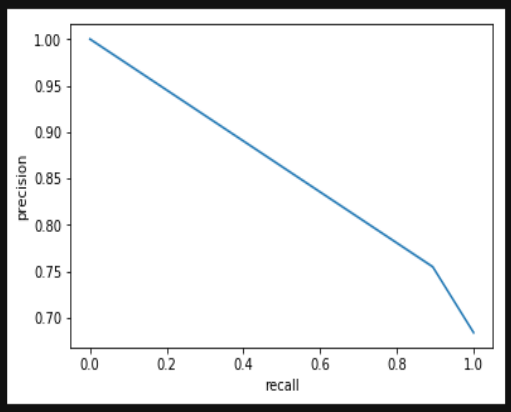
[234]:
```
prec,recall,threshhold=metrics.precision_recall_curve(ytest,predict_test)
plt.plot(threshhold,prec[:-1],'b--')
plt.plot(threshhold,recall[:-1],'g-')
```

[234]: [<matplotlib.lines.Line2D at 0x7f46ac3c4400>]



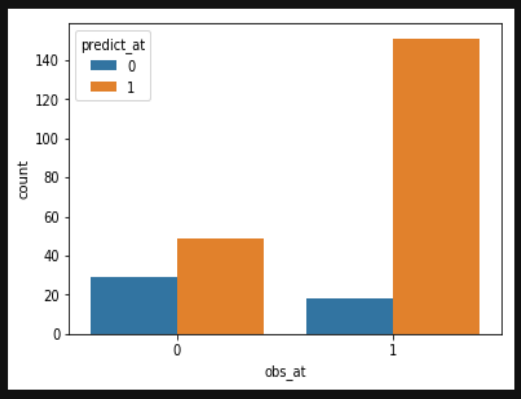
[235]:
```
plt.plot(recall,prec)
plt.xlabel("recall")
plt.ylabel("precision")
```

[235]: Text(0, 0.5, 'precision')



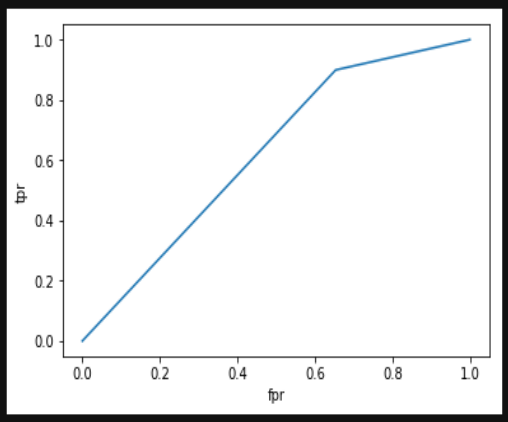
[236]:
```
re = list(np.arange(1,248,1))
da = {'record':re,'obs_at':list(ytest),'predict_at':predict_test.tolist()}
da = pd.DataFrame(da)
sns.countplot(x='obs_at',hue='predict_at',data=da)
```

[236]: <matplotlib.axes._subplots.AxesSubplot at 0x7f46ac35c358>



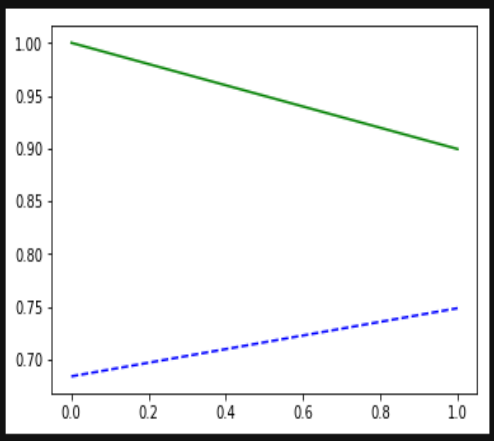
[237]:
```
tmodel=0
model=linear_model.LogisticRegression()
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    modelcvscore=model_selection.cross_val_score(model,X,y,cv=5,scoring=score)
    if score=="f1":
        print("f1-score",":",round(modelcvscore.mean()*100))
    elif score=="roc_auc":
        print("AUC",":",round(modelcvscore.mean()*100))
    else:
        print(score,":",round(modelcvscore.mean()*100))
```

[239]: Text(0, 0.5, 'tpr')

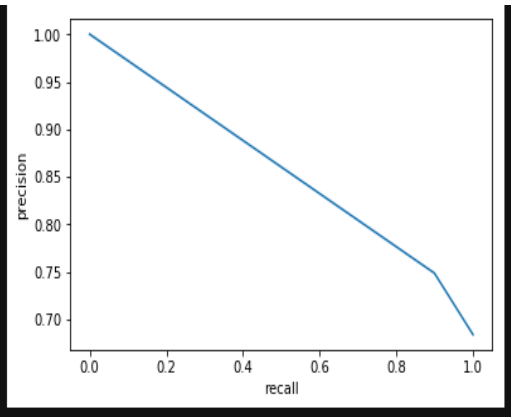


[240]:
```
prec,recall,threshhold=metrics.precision_recall_curve(ytest,predict_test)
plt.plot(threshhold,prec[:-1],'b--')
plt.plot(threshhold,recall[:-1],'g-')
```

[240]: [<matplotlib.lines.Line2D at 0x7f46ac241128>]



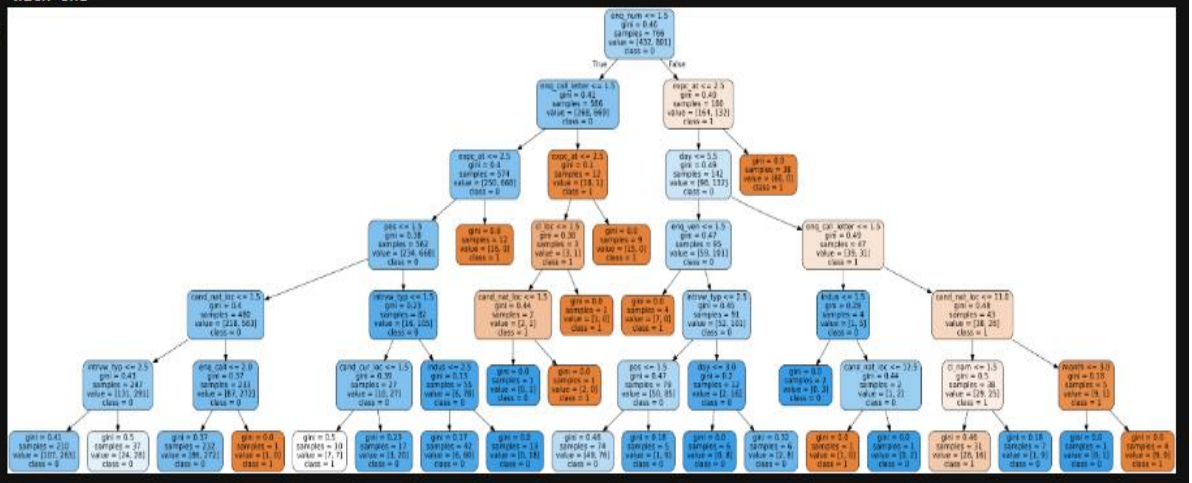
[241]:
```
plt.plot(recall,prec)
plt.xlabel("recall")
plt.ylabel("precision")
```



[242]:
```
tmodel=0
model=naive_bayes.BernoulliNB()
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    modelcvscore=model_selection.cross_val_score(model,X,y,cv=5,scoring=score)
    if score=="f1":
        print("f1-score",":",round(modelcvscore.mean()*100))
    elif score=="roc_auc":
        print("AUC",":",round(modelcvscore.mean()*100))
    else:
        print(score,":",round(modelcvscore.mean()*100))
```

with ohe

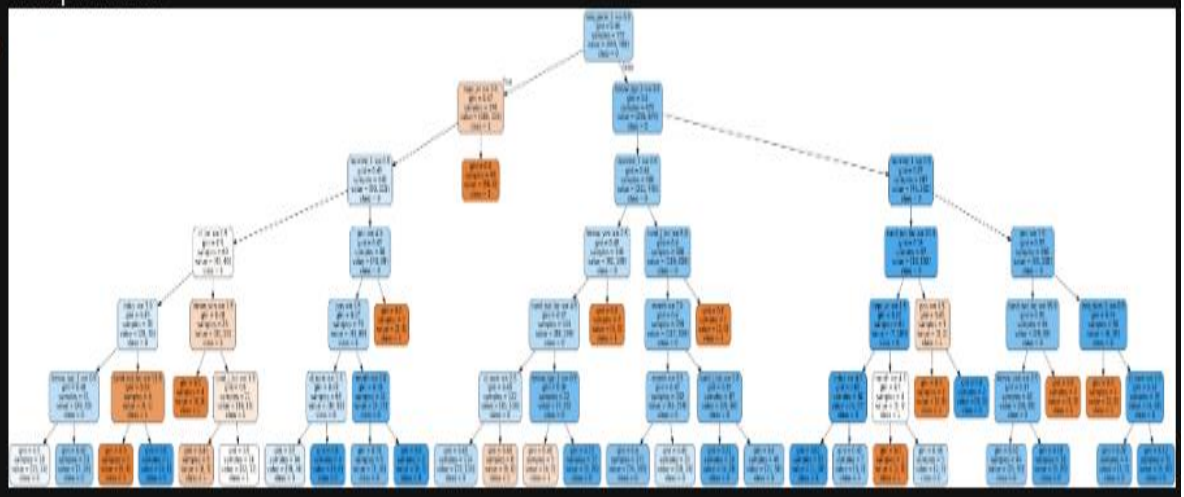
[225]:



[226]:
```
print("with partial ohe")
Image(filename = 'tree3.png')
```

with partial ohe

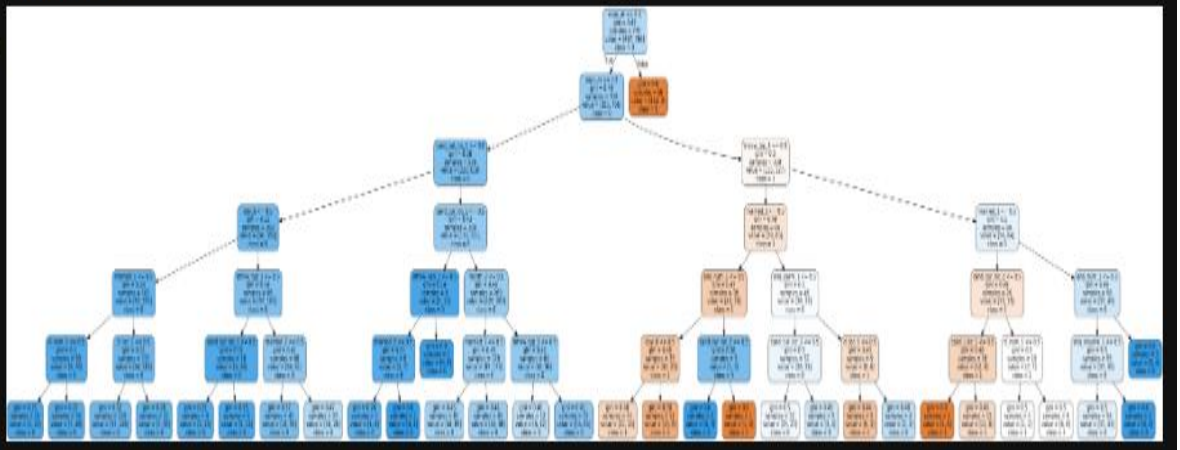
[226]:



[227]:
```
print("with full ohe")
Image(filename = 'tree4.png')
```

with full ohe

[227]:



User Defined Functions Used:

```
def printresult(actual,predicted):
    confmatrix=metrics.confusion_matrix(actual,predicted)
    accscore=metrics.accuracy_score(actual,predicted)
    precscore=metrics.precision_score(actual,predicted)
    recscore=metrics.recall_score(actual,predicted)
    print(confmatrix)
    print("accuracy : {:.4f}".format(accscore))
    print("precision : {:.4f}".format(precscore))
    print("recall : {:.4f}".format(recscore))
    print("f1-score : {:.4f}".format(metrics.f1_score(actual,predicted)))
    print("AUC : {:.4f}".format(metrics.roc_auc_score(actual,predicted)))
```

Here a function is defined named “printresult”. In it we pass actual and predicted result and gives as output confusion matrix and scores of accuracy, precision,recall,f1-score and auc.

DATA CLEANING

Data cleaning or **data cleansing** is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.[1] Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting.

After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores. Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

The actual process of data cleansing may involve removing typographical errors or validating and correcting values against a known list of entities. The validation may be strict (such as rejecting any address that does not have a valid postal code) or fuzzy (such as correcting records that partially match existing, known records). Some data cleansing solutions will clean data by cross checking with a validated data set. A common data cleansing practice is data enhancement, where data is made more complete by adding related information. For example, appending addresses with any phone numbers related to that address. Data cleansing may also involve

activities like, harmonization of data, and standardization of data. For example, harmonization of short codes (st, rd, etc.) to actual words (street, road, etcetera). Standardization of data is a means of changing a reference data set to a new standard, ex, use of standard codes.

Data cleaning can also be defined as the process of altering data in a given storage resource to make sure that it is accurate and correct. There are many ways to pursue data cleaning in various software and data storage architectures; most of them center on the careful review of data sets and the protocols associated with any particular data storage technology.

Data cleansing is sometimes compared to data purging, where old or useless data will be deleted from a data set. Although data cleansing can involve deleting old, incomplete or duplicated data, data cleansing is different from data purging in that data purging usually focuses on clearing space for new data, whereas data cleansing focuses on maximizing the accuracy of data in a system. A data cleansing method may use parsing or other methods to get rid of syntax errors, typographical errors or fragments of records. Careful analysis of a data set can show how merging multiple sets led to duplication, in which case data cleansing may be used to fix the problem.

Many issues involving data cleansing are similar to problems that archivists, database admin staff and others face around processes like data maintenance, targeted data mining and the extract, transform, load (ETL) methodology, where old data is reloaded into a new data set. These issues often regard the syntax and specific use of command to effect related tasks in database and server technologies like SQL or Oracle. Database administration is a highly important role in many businesses and

organizations that rely on large data sets and accurate records for commerce or any other initiative.

The data cleaning was done by the below mentioned processes:-

1. Renaming column names

Column	Renamed
Date of Interview	doi
Client name	cl_nam
Industry	indus
Location	cl_loc
Position to be closed	pos
Nature of Skillset	skill
Interview Type	intrvw_typ
Name(Cand ID)	cand_nam
Gender	gend
Candidate Current Location	cand_cur_loc

Candidate Job Location	cand_j_loc
Interview Venue	intrvw_ven
Candidate Native location	cand_nat_loc
Have you obtained the necessary permission to start at the required time	enq_perm
Hope there will be no unscheduled meetings	enq_unsch_meet
Can I Call you three hours before the interview and follow up on your attendance for the interview	enq_call
Can I have an alternative number/ desk number. I assure you that I will not trouble you too much	enq_num
Have you taken a printout of your updated resume. Have you read the JD and understood the same	enq_resume
Are you clear with the venue details and the landmark.	enq_ven
Has the call letter been shared	enq_call_letter
Expected Attendance	expc_at

Observed Attendance	obs_at
Marital Status	married

The data sheet have 1234 entries, where 28 columns are present. We renamed the column names to make it look more accurate.

For example,

Column	**Renamed**
Date of interview	doi
Client name	cl_nam
Industry	indus
Location	cl_loc
Can I call you three hours before the interview and follow up on your attendance for the interview	enq_call

The Date of interview is renamed to “doi”, Client name to “cl_nam”, Industry to “indus”, Location to “cl_loc” i.e, client location . The columns having question names like Can I call you three hours before were renamed as an enquiry call as “enq_call” and so on.

2. Removing spaces

   Removing of unnecessary spaces have been done while the cleansing of data.
   This was done using the str.split() function which is an inbuilt function in python.
3. Unnamed columns dropped

   The columns that are unnamed in the data sheet are dropped. There are total five unnamed columns.
   Unnamed 23, unnamed 24, unnamed 25, unnamed 26 and unnamed 27.

Before:

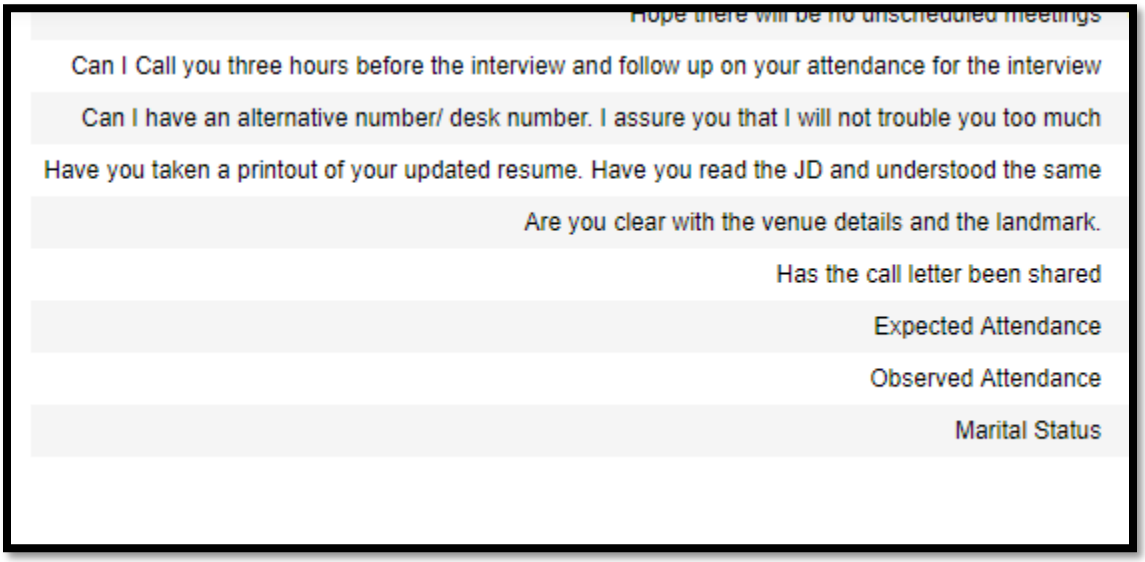
Unnamed: 23	Unnamed: 24	Unnamed: 25	Unnamed: 26	Unnamed: 27
NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN

Code:

```
print(df.columns[-5:])
df[df.columns[-5:]].describe()
df.drop(df.columns[-5:],axis=1,inplace=True)
```

```
Index(['Unnamed: 23', 'Unnamed: 24', 'Unnamed: 25', 'Unnamed: 26',
       'Unnamed: 27'],
      dtype='object')
```

Output:



4. Handling of null value

All the columns were checked and we have searched for null values if any. We found the total number of null datas. Then we handled those datas by replacing the null values with some other values mostly with a “no” in our project. In the end there were no null values and the dataset was ready to be modeled.

Before:

For some columns total sum value was calculated-

```
[40]: #df_temp=df1.copy
      df1.enq_perm.isna().sum()

[40]: 205
```

```
[44]: df1.enq_unsch_meet.isna().sum()

[44]: 247
```

After:

```
[85]: for col in df1.columns:
          print(col," ",df1[col].isna().sum())

cl_nam   0
indus   0
cl_loc   0
pos   0
intrvw_typ   0
cand_nam   0
gend   0
cand_cur_loc   0
cand_j_loc   0
intrvw_ven   0
cand_nat_loc   0
enq_perm   0
enq_unsch_meet   0
enq_call   0
enq_num   0
enq_resume   0
enq_ven   0
enq_call_letter   0
expc_at   0
obs_at   0
married   0
month   0
day   0
```

```
[87]: df1.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1233 entries, 0 to 1232
Data columns (total 23 columns):
cl_nam             1233 non-null object
indus              1233 non-null object
cl_loc             1233 non-null object
pos                1233 non-null object
intrvw_typ         1233 non-null object
cand_nam           1233 non-null object
gend               1233 non-null object
cand_cur_loc       1233 non-null object
cand_j_loc         1233 non-null object
intrvw_ven         1233 non-null object
cand_nat_loc       1233 non-null object
enq_perm           1233 non-null object
enq_unsch_meet     1233 non-null object
enq_call           1233 non-null object
enq_num            1233 non-null object
enq_resume         1233 non-null object
enq_ven            1233 non-null object
enq_call_letter    1233 non-null object
expc_at            1233 non-null object
obs_at             1233 non-null object
married            1233 non-null object
month              1233 non-null int64
day                1233 non-null int64
dtypes: int64(2), object(21)
memory usage: 271.2+ KB
```

5. Categorising

Before:

```
In [9]: df1.cl_nam.value_counts()

Out[9]: Standard Chartered Bank             904
        Hospira                              75
        Pfizer                               75
        Aon Hewitt                           28
        Flextronics                          23
        ANZ                                  22
        Hewitt                               20
        UST                                  18
        Standard Chartered Bank Chennai      17
        Prodapt                              17
        Astrazeneca                          15
        Williams Lea                         11
        Barclays                              5
        Aon hewitt Gurgaon                    2
        Woori Bank                            1
                                              1
        Name: cl_nam, dtype: int64
```

After:

```
In [10]: l1=[
             "Standard Chartered Bank Chennai",
             "Hewitt",
             "Aon hewitt Gurgaon"
         ]

         l2=[
             "standard chartered bank",
             "aon hewitt",
             "aon hewitt",
         ]

In [11]: df1.cl_nam.replace(l1,l2,inplace=True)
         df1.cl_nam=df1.cl_nam.str.lower()
         df1.cl_nam.value_counts()
```

Output:

```
In [11]: df1.cl_nam.replace(l1,l2,inplace=True)
         df1.cl_nam=df1.cl_nam.str.lower()
         df1.cl_nam.value_counts()

Out[11]: standard chartered bank    921
         hospira                     75
         pfizer                      75
         aon hewitt                  50
         flextronics                 23
         anz                         22
         ust                         18
         prodapt                     17
         astrazeneca                 15
         williams lea                11
         barclays                     5
         woori bank                   1
                                      1
         Name: cl_nam, dtype: int64
```

Before:

Industry :

```
In [13]: df1.indus.value_counts()

Out[13]: BFSI                        949
         Pharmaceuticals             165
         IT Products and Services     45
         Electronics                  23
         IT Services                  23
         Telecom                      17
         IT                           11
         Name: indus, dtype: int64
```

After:

```
df1.indus.replace(["IT Products and Services","IT Services"],["IT","IT"],inplace=True)
df1.indus=df1.indus.str.lower()
print(df1.indus.value_counts())
```

```
bfsi               949
pharmaceuticals    165
it                  79
electronics         23
telecom             17
Name: indus, dtype: int64
```

Before:

Client Location:

```
In [16]: print(df1.cl_loc.value_counts())
```

```
Chennai       754
Bangalore     292
chennai        86
Hyderabad      38
Gurgaon        33
Noida          15
- Cochin-       9
chennai         3
Delhi           1
Gurgaonr        1
CHENNAI         1
Name: cl_loc, dtype: int64
```

After:

```
df1.cl_loc=df1.cl_loc.str.lower().str.strip()
df1.cl_loc.replace(["gurgaonr","- cochin-"],["gurgaon","cochin"],inplace=True)
print(df1.cl_loc.value_counts())
```

```
chennai      844
bangalore    292
hyderabad     38
gurgaon       34
noida         15
cochin         9
delhi          1
Name: cl_loc, dtype: int64
```

Before:

Position to be closed:

```
In [19]: print(df1.pos.value_counts())
```

```
Routine                1023
Niche                   163
Dot Net                  18
Trade Finance            11
AML                       8
Selenium testing          5
Production- Sterile       5
Name: pos, dtype: int64
```

After:

```
In [20]: df1.pos=df1.pos.str.lower()
         print(df1.pos.value_counts())

routine                1023
niche                   163
dot net                  18
trade finance            11
aml                       8
selenium testing          5
production- sterile       5
Name: pos, dtype: int64
```

Before:

Interview type:

```
In [22]: df1.intrvw_typ.value_counts()

Out[22]: Scheduled Walk In    456
         Scheduled            371
         Walkin               189
         Scheduled Walkin     189
         Walkin                27
         Sceduled walkin        1
         Name: intrvw_typ, dtype: int64
```

After:

```
df1.intrvw_typ=df1.intrvw_typ.str.lower().str.strip()
df1.intrvw_typ.replace(["scheduled walk in","sceduled walkin"],["scheduled walkin","s
df1.intrvw_typ.value_counts()

scheduled walkin    646
scheduled           371
walkin              216
Name: intrvw_typ, dtype: int64
```

Before:

Candidate current location:

```
df1.cand_cur_loc.value_counts()
```

```
Chennai       754
Bangalore     292
chennai        86
Hyderabad      38
Gurgaon        34
Noida          15
- Cochin-       9
chennai         3
Delhi           1
CHENNAI         1
Name: cand_cur_loc, dtype: int64
```

After:

```
df1.cand_cur_loc=df1.cand_cur_loc.str.lower().str.strip()
df1.cand_cur_loc.replace(["- cochin-"],["cochin"],inplace=True)
print(df1.cand_cur_loc.value_counts())
```

```
chennai      844
bangalore    292
hyderabad     38
gurgaon       34
noida         15
cochin         9
delhi          1
Name: cand_cur_loc, dtype: int64
```

Candidate Job locations:

Before:

```
df1.cand_j_loc.value_counts()
```

```
Chennai          893
Bangalore        259
Gurgaon           35
Visakapatinam     21
Noida             15
- Cochin-          9
Hosur              1
Name: cand_j_loc, dtype: int64
```

After:

```
df1.cand_j_loc=df1.cand_cur_loc.str.lower().str.strip()
df1.cand_j_loc.replace(["- cochin-"],["cochin"],inplace=True)
print(df1.cand_j_loc.value_counts())
```

```
chennai      844
bangalore    292
hyderabad     38
gurgaon       34
noida         15
cochin         9
delhi          1
Name: cand_j_loc, dtype: int64
```

Interview venue:

Before:

```
df1.intrvw_ven.value_counts()
```

```
Chennai       852
Bangalore     277
Hyderabad      40
Gurgaon        35
Noida          15
- Cochin-       9
Hosur           5
Name: intrvw_ven, dtype: int64
```

After:

```
df1.intrvw_ven=df1.intrvw_ven.str.lower().str.strip()
df1.intrvw_ven.replace(["- cochin-"],["cochin"],inplace=True)
print(df1.intrvw_ven.value_counts())
```

```
chennai      852
bangalore    277
hyderabad     40
gurgaon       35
noida         15
cochin         9
hosur          5
Name: intrvw_ven, dtype: int64
```

Have you obtained the necessary permission to start at the required time

Before:

```
df1.enq_perm.value_counts()
```

```
Yes               917
No                 79
Not yet            19
Na                  5
yes                 4
Yet to confirm      4
NO                  1
Name: enq_perm, dtype: int64
```

After:

```
df1.enq_perm=df1.enq_perm.str.lower()
df1.enq_perm.replace(["not yet","na","yet to confirm"],["no","no","no"],inplace=True)
print(df1.enq_perm.value_counts())
```

```
yes    921
no     108
Name: enq_perm, dtype: int64
```

Hope there will be no unscheduled meetings

Before:

```
df1.enq_unsch_meet.str.lower().value_counts()
```

```
yes         954
na           20
no            6
not sure      5
cant say      1
Name: enq_unsch_meet, dtype: int64
```

After:

```
df1.enq_unsch_meet=df1.enq_unsch_meet.str.lower()
df1.enq_unsch_meet.replace(["na","cant say"],["no","not sure"],inplace=True)
df1.enq_unsch_meet.value_counts()
```

```
yes         954
no           26
not sure      6
Name: enq_unsch_meet, dtype: int64
```

Can I Call you three hours before the interview and follow up on your attendance for the interview

Before:

```
df1.enq_call.value_counts()
```

```
Yes        951
Na          20
No          10
yes          4
No Dont      1
Name: enq_call, dtype: int64
```

After:

```
df1.enq_call=df1.enq_call.str.lower()
df1.enq_call.replace(["na","no dont"],["no","no"],inplace=True)
df1.enq_call.value_counts()
```

```
yes    955
no      31
Name: enq_call, dtype: int64
```

Can I have an alternative number/ desk number. I assure you that I will not trouble you too much

Before:

```
df1.enq_num.value_counts()
```

```
Yes                          936
No                            27
Na                            19
No I have only thi number      2
yes                            1
na                             1
Name: enq_num, dtype: int64
```

After:

```
df1.enq_num=df1.enq_num.str.lower()
df1.enq_num.replace(["na","no i have only thi number"],["no","no"],inplace=True)
print(df1.enq_num.value_counts())
```

```
yes    937
no      49
Name: enq_num, dtype: int64
```

Have you taken a printout of your updated resume. Have you read the JD and understood the same

Before:

```
df1.enq_resume.value_counts()
```

```
Yes                       940
Na                         19
No                         16
Not Yet                     4
Not yet                     2
yes                         2
No- will take it soon       1
na                          1
Name: enq_resume, dtype: int64
```

After:

```
df1.enq_resume=df1.enq_resume.str.lower()
df1.enq_resume.replace(["na","not yet","no- will take it soon"],["no","no","no"],inplace=True)
df1.enq_resume.value_counts()
```

```
yes    942
no      43
Name: enq_resume, dtype: int64
```

Have you taken a printout of your updated resume. Have you read the JD and understood the same

Before:

```
df1.enq_resume.value_counts()
```

```
Yes                       940
Na                         19
No                         16
Not Yet                     4
Not yet                     2
yes                         2
No- will take it soon       1
na                          1
Name: enq_resume, dtype: int64
```

After:

```
df1.enq_resume=df1.enq_resume.str.lower()
df1.enq_resume.replace(["na","not yet","no- will take it soon"],["no","no","no"],inplace=True)
df1.enq_resume.value_counts()
```

```
yes    942
no      43
Name: enq_resume, dtype: int64
```

Are you clear with the venue details and the landmark.

Before:

```
df1.enq_ven.value_counts()
```

```
Yes                    946
Na                      19
No                      14
yes                      2
No- I need to check      2
no                       1
na                       1
Name: enq_ven, dtype: int64
```

After:

```
df1.enq_ven=df1.enq_ven.str.lower()
df1.enq_ven.replace(["na","no- i need to check"],["no","no"],inplace=True)
print(df1.enq_ven.value_counts())
```

```
yes    948
no      37
Name: enq_ven, dtype: int64
```

Has the call letter been shared

Before:

```
df1.enq_call_letter.value_counts()
```

```
Yes                932
Na                  19
No                  17
Not Sure             8
Need To Check        3
Not yet              2
yes                  2
Not sure             1
Havent Checked       1
Yet to Check         1
na                   1
no                   1
Name: enq_call_letter, dtype: int64
```

After:

```
df1.enq_call_letter=df1.enq_call_letter.str.lower()
df1.enq_call_letter.replace(["na","not yet","not sure","havent checked","yet to check"],["no","no","need to check","need to check
print(df1.enq_call_letter.value_counts())
```

```
yes              934
no                40
need to check     14
Name: enq_call_letter, dtype: int64
```

```
_letter.str.lower()
","not yet","not sure","havent checked","yet to check"],["no","no","need to check","need to check","need to check"],inplace=True)
counts())
```

```
yes             934
no               40
need to check    14
Name: enq_call_letter, dtype: int64
```

Expected Attendance

Before:

```
df1.expc_at.value_counts()
```

```
Yes          882
Uncertain    250
No            59
NO            34
10.30 Am       1
yes            1
11:00 AM       1
Name: expc_at, dtype: int64
```

After:

```
df1.expc_at=df1.expc_at.str.lower()
df1.expc_at.replace(["11:00 am","10.30 am"],["yes","yes"],inplace=True)
print(df1.expc_at.value_counts())
```

```
yes          885
uncertain    250
no            93
Name: expc_at, dtype: int64
```

Observed Attendance

Before:

```
df1.obs_at.value_counts()
```

```
Yes     701
No      401
yes      81
NO       35
no        7
No        6
yes       1
no        1
Name: obs_at, dtype: int64
```

After:

```
df1.obs_at=df1.obs_at.str.lower().str.strip()
print(df1.obs_at.value_counts())
```

```
yes    783
no     450
Name: obs_at, dtype: int64
```

Marital Status

Before:

```
df1.married.value_counts()
```

```
Single     767
Married    466
Name: married, dtype: int64
```

After:

```
df1.married=df1.married.str.lower()
print(df1.married.value_counts())
```

```
single     767
married    466
Name: married, dtype: int64
```

ONE HOT ENCODING

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. Our categories were formerly rows, but now they're columns. Our numerical variable, calories, has however stayed the same. A 1 in a particular column will tell the computer the correct category for that row's data. In other words, we have created an additional binary column for each category. The only disadvantage of ONE is the number of columns increases immensely. We first one hot encode few columns and then we encode all the columns.

```
[90]: df2=df1.copy()
      def t(col):
          j=1
          dic={}
          for i in df2[col].value_counts().index:
              dic[i]=j
              j+=1
          df2[col].replace(dic,inplace=True)

[91]: for col in df2.columns:
          if(col=='month' or col=='day'):
              continue
          t(col)

[92]: df3 = df2.copy()
      df4=df2.copy()

[93]: ohel_half = ['intrvw_typ','gend','enq_perm','enq_unsch_meet','enq_call','enq_num','enq_resume','enq_ven','enq_call_letter','married']
      ohel_full=['cl_nam', 'indus', 'cl_loc', 'pos', 'intrvw_typ', 'gend',
             'cand_cur_loc', 'cand_j_loc', 'intrvw_ven', 'cand_nat_loc', 'enq_perm',
             'enq_unsch_meet', 'enq_call', 'enq_num', 'enq_resume', 'enq_ven',
             'enq_call_letter', 'married', 'month', 'day']
      df3 = pd.get_dummies(df3, prefix=ohel_half, columns=ohel_half)
      df4 = pd.get_dummies(df4,prefix=ohel_full,columns=ohel_full)

[94]: print(len(df3.columns))
      print(len(df4.columns))

      40
      145
```

After doing one hot encoding to all columns there will be 145 columns in total.

Now from these 145 columns we have to choose the best features that contribute most to the dataset prediction. For this we use some feature selection functions.

```
[97]: print("\nwithout ohe")
      x=df2.drop("obs_at",axis=1)
      y=df2.obs_at
      bestfeatures = SelectKBest(score_func=chi2, k=10)
      fit = bestfeatures.fit(x,y)
      dfscores = pd.DataFrame(fit.scores_)
      dfcolumns = pd.DataFrame(x.columns)
      featureScores = pd.concat([dfcolumns,dfscores],axis=1)
      featureScores.columns = ['Specs','Score']  #naming the dataframe columns
      print(featureScores.nlargest(8,'Score'))  #print 8 best features
      col_df2=featureScores.nlargest(8,"Score")["Specs"].tolist()

      print("\n\nafter half ohe")
      x=df3.drop("obs_at",axis=1)
      y=df3.obs_at
      bestfeatures = SelectKBest(score_func=chi2, k=10)
      fit = bestfeatures.fit(x,y)
      dfscores = pd.DataFrame(fit.scores_)
      dfcolumns = pd.DataFrame(x.columns)
      featureScores = pd.concat([dfcolumns,dfscores],axis=1)
      featureScores.columns = ['Specs','Score']  #naming the dataframe columns
      print(featureScores.nlargest(8,'Score'))  #print 8 best features
      col_df3=featureScores.nlargest(8,"Score")["Specs"].tolist()

      print("\n\nafter full ohe")
      x=df4.drop("obs_at",axis=1)
      y=df4.obs_at
      bestfeatures = SelectKBest(score_func=chi2, k=10)
      fit = bestfeatures.fit(x,y)
      dfscores = pd.DataFrame(fit.scores_)
      dfcolumns = pd.DataFrame(x.columns)
      featureScores = pd.concat([dfcolumns,dfscores],axis=1)
      featureScores.columns = ['Specs','Score']  #naming the dataframe columns
      print(featureScores.nlargest(8,'Score'))
      col_df4=featureScores.nlargest(8,"Score")["Specs"].tolist()

      without ohe
                   Specs      Score
      17         expc_at  58.032965
      10        enq_perm  50.952692
      16  enq_call_letter  24.395756
      14       enq_resume  23.867266
      13          enq_num  22.686281
      15          enq_ven  20.275348
      11   enq_unsch_meet  18.719412
      12         enq_call  15.416396
```

```
after half ohe
              Specs       Score
18        enq_perm_3  127.912950
8            expc_at   58.032965
35  enq_call_letter_2   51.267031
20   enq_unsch_meet_2   50.960993
30       enq_resume_3   45.552318
23         enq_call_2   45.175754
26          enq_num_2   45.175754
29       enq_resume_2   44.350101


after full ohe
              Specs       Score
106       enq_perm_3  127.912950
0            expc_at   58.032965
123 enq_call_letter_2   51.267031
108  enq_unsch_meet_2   50.960993
118      enq_resume_3   45.552318
111        enq_call_2   45.175754
114         enq_num_2   45.175754
117      enq_resume_2   44.350101
```

The top 8 best features are selected using kbest selection method. Now we fit the columns to different models.

MODEL BUILDING

We apply different models:

1. Logistic Regression:
   Logistic regression is a statistical **model** that in its basic form uses a **logistic** function to **model** a binary dependent variable, although many more complex extensions exist. In **regression** analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a **logistic model** (a form of binary**regression**).
   We take one column (obs_data) for test and rest all columns for training. We use variance threshold for feature selection. We fit Xtrain and ytrain into linear model. Then we predict the result using Xtest. Then we use a user defined function “printresult” to print the confusion matrix and scores of accuracy,precision,recall,f1-score,auc_roc.

Feature selection:

[228]:
```
X = df3.drop(['obs_at','expc_at'],axis=1)
y = df3['obs_at']
thresholder = VarianceThreshold(threshold=.22)
thresholder.fit_transform(X)
temp = X.columns[thresholder.get_support()]
temp
```

[228]:
```
Index(['cl_nam', 'indus', 'cl_loc', 'pos', 'cand_cur_loc', 'cand_j_loc',
       'intrvw_ven', 'cand_nat_loc', 'month', 'day', 'intrvw_typ_1',
       'married_1', 'married_2'],
      dtype='object')
```

[229]:
```
feature_scores = feature_selection.mutual_info_classif(X, y)
micc = []
for score, fname in sorted(zip(feature_scores, X.columns.tolist()), reverse=True)[:10]:
    micc += [fname]
micc
```

[229]:
```
['enq_perm_3',
 'enq_call_letter_1',
 'enq_unsch_meet_2',
 'enq_call_1',
 'enq_unsch_meet_1',
 'enq_call_letter_2',
 'enq_perm_1',
 'enq_num_1',
 'enq_resume_1',
 'enq_ven_2']
```

[230]:
```
X2 = X[micc]
thresholder2 = VarianceThreshold(threshold=.18)
thresholder2.fit_transform(X2)
temp2 = X2.columns[thresholder2.get_support()]
temp2
```

[230]:
```
Index(['enq_call_letter_1', 'enq_perm_1', 'enq_num_1', 'enq_resume_1'], dtype='object')
```

[231]:
```
X = X[temp2.tolist()+temp.tolist()]
len(X.columns)
```

[231]:
```
17
```

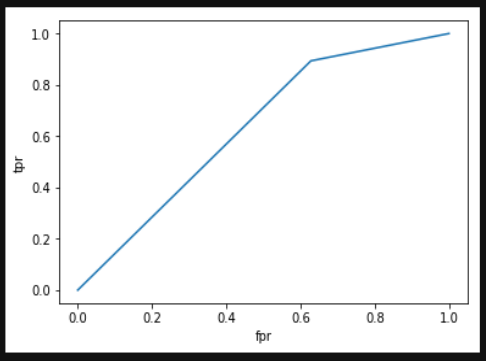
[232]:
```
Xtrain,Xtest,ytrain,ytest=model_selection.train_test_split(X,y,test_size=.2,random_state=21)
model=linear_model.LogisticRegression()
model.fit(Xtrain,ytrain)
predict_test=model.predict(Xtest)
printresult(ytest,predict_test)
```

```
[[ 29  49]
 [ 18 151]]
accuracy : 0.7287
precision : 0.7550
recall : 0.8935
f1-score : 0.8184
AUC : 0.6326
```

```
/home/soham/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver will b
e changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

[233]:
```
tpr,fpr,threshhold=metrics.roc_curve(ytest,predict_test)
plt.plot(tpr,fpr)
plt.xlabel("fpr")
plt.ylabel("tpr")
```

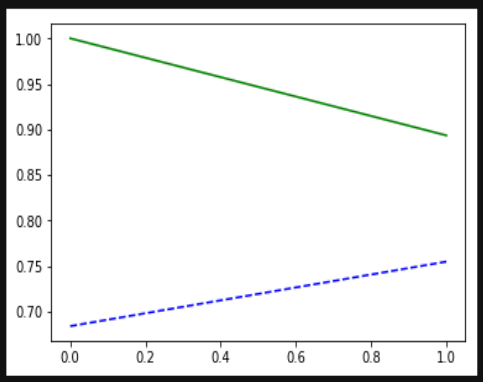
[233]:
```
Text(0, 0.5, 'tpr')
```



[234]:
```
prec,recall,threshhold=metrics.precision_recall_curve(ytest,predict_test)
plt.plot(threshhold,prec[:-1],'b--')
plt.plot(threshhold,recall[:-1],'g-')
```

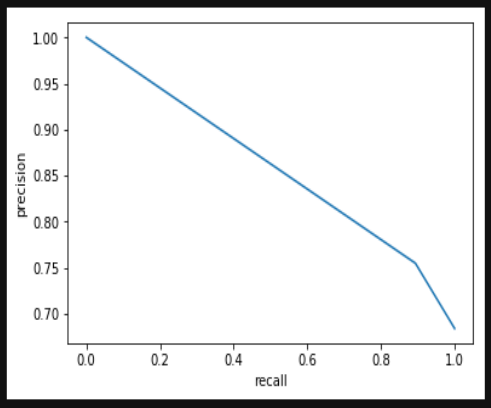

[234]:
```
prec,recall,threshhold=metrics.precision_recall_curve(ytest,predict_test)
plt.plot(threshhold,prec[:-1],'b--')
plt.plot(threshhold,recall[:-1],'g-')
```


[234]:
```
[<matplotlib.lines.Line2D at 0x7f46ac3c4400>]
```



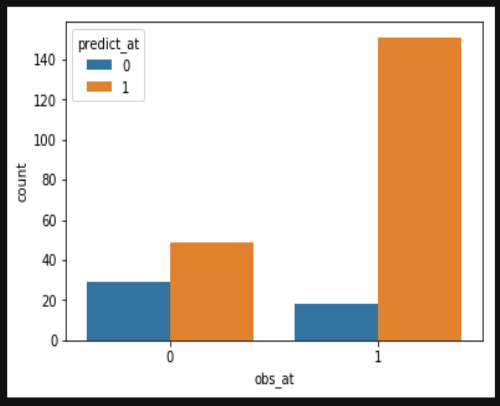
[235]:
```
plt.plot(recall,prec)
plt.xlabel("recall")
plt.ylabel("precision")
```

[235]: Text(0, 0.5, 'precision')



[236]:
```
re = list(np.arange(1,248,1))
da = {'record':re,'obs_at':list(ytest),'predict_at':predict_test.tolist()}
da = pd.DataFrame(da)
sns.countplot(x='obs_at',hue='predict_at',data=da)
```

[236]: <matplotlib.axes._subplots.AxesSubplot at 0x7f46ac35c358>



[237]:
```
tmodel=0
model=linear_model.LogisticRegression()
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    modelcvscore=model_selection.cross_val_score(model,X,y,cv=5,scoring=score)
    if score=="f1":
        print("f1-score",":",round(modelcvscore.mean()*100))
    elif score=="roc_auc":
        print("AUC",":",round(modelcvscore.mean()*100))
    else:
        print(score,":",round(modelcvscore.mean()*100))
```

Results:

Scoring	Scores
accuracy	68.0
precision	72.0
recall	85.0
f1-score	77.0
AUC	62.0

Here after logistic regression model we see precision is 72% and recall is 85%.

2.Naive Bayes:

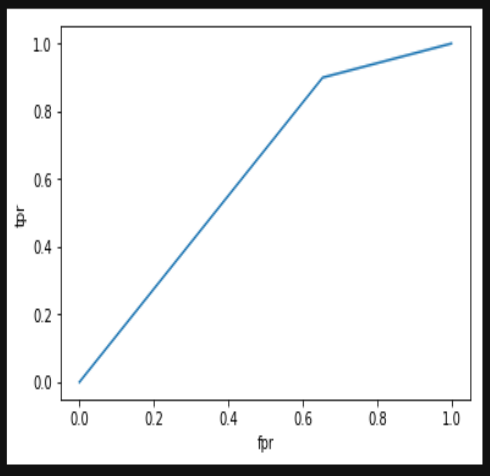
In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the text retrieval community in the early 1960s and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines.It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression,[4]:718 which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

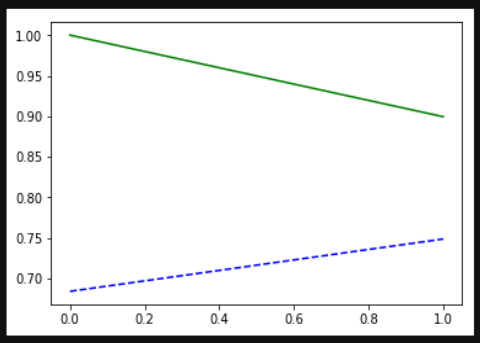
In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.

```
[238]: model=naive_bayes.BernoulliNB()
       model.fit(Xtrain,ytrain)
       predict_test=model.predict(Xtest)
       printresult(ytest,predict_test)

       [[ 27  51]
        [ 17 152]]
       accuracy : 0.7247
       precision : 0.7488
       recall : 0.8994
       f1-score : 0.8172
       AUC : 0.6228

[239]: tpr,fpr,threshhold=metrics.roc_curve(ytest,predict_test)
       plt.plot(tpr,fpr)
       plt.xlabel("fpr")
       plt.ylabel("tpr")

[239]: Text(0, 0.5, 'tpr')
```

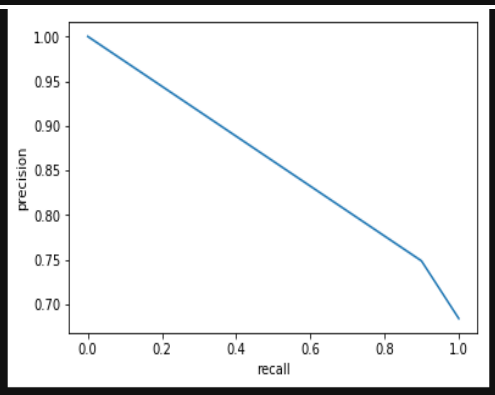



```
[240]: prec,recall,threshhold=metrics.precision_recall_curve(ytest,predict_test)
       plt.plot(threshhold,prec[:-1],'b--')
       plt.plot(threshhold,recall[:-1],'g-')
```

```
[240]: [<matplotlib.lines.Line2D at 0x7f46ac241128>]
```




```
[241]: plt.plot(recall,prec)
       plt.xlabel("recall")
       plt.ylabel("precision")
```




```
[242]: tmodel=0
model=naive_bayes.BernoulliNB()
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    modelcvscore=model_selection.cross_val_score(model,X,y,cv=5,scoring=score)
    if score=="f1":
        print("f1-score",":",round(modelcvscore.mean()*100))
    elif score=="roc_auc":
        print("AUC",":",round(modelcvscore.mean()*100))
    else:
        print(score,":",round(modelcvscore.mean()*100))
```

Results:

Scoring	Scores
accuracy	70.0
precision	72.0
recall	86.0
f1-score	78.0
AUC	64.0

After applying the naive bayes model we see precision is 72% and recall is 86%.

3.Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The first algorithm for random decision forests was created by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

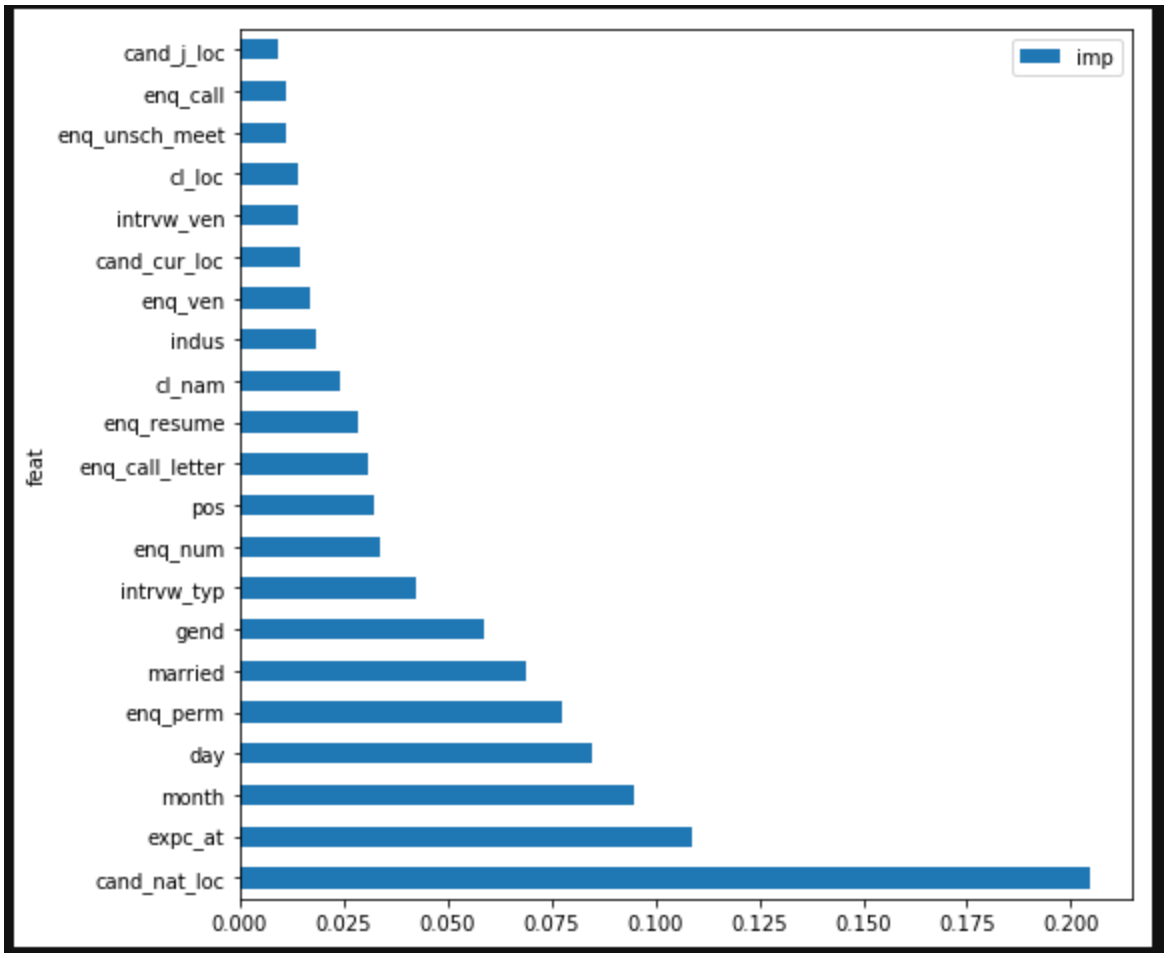
An extension of the algorithm was developed by Leo Breiman and Adele Cutler,who registered "Random Forests" as a trademark (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance.

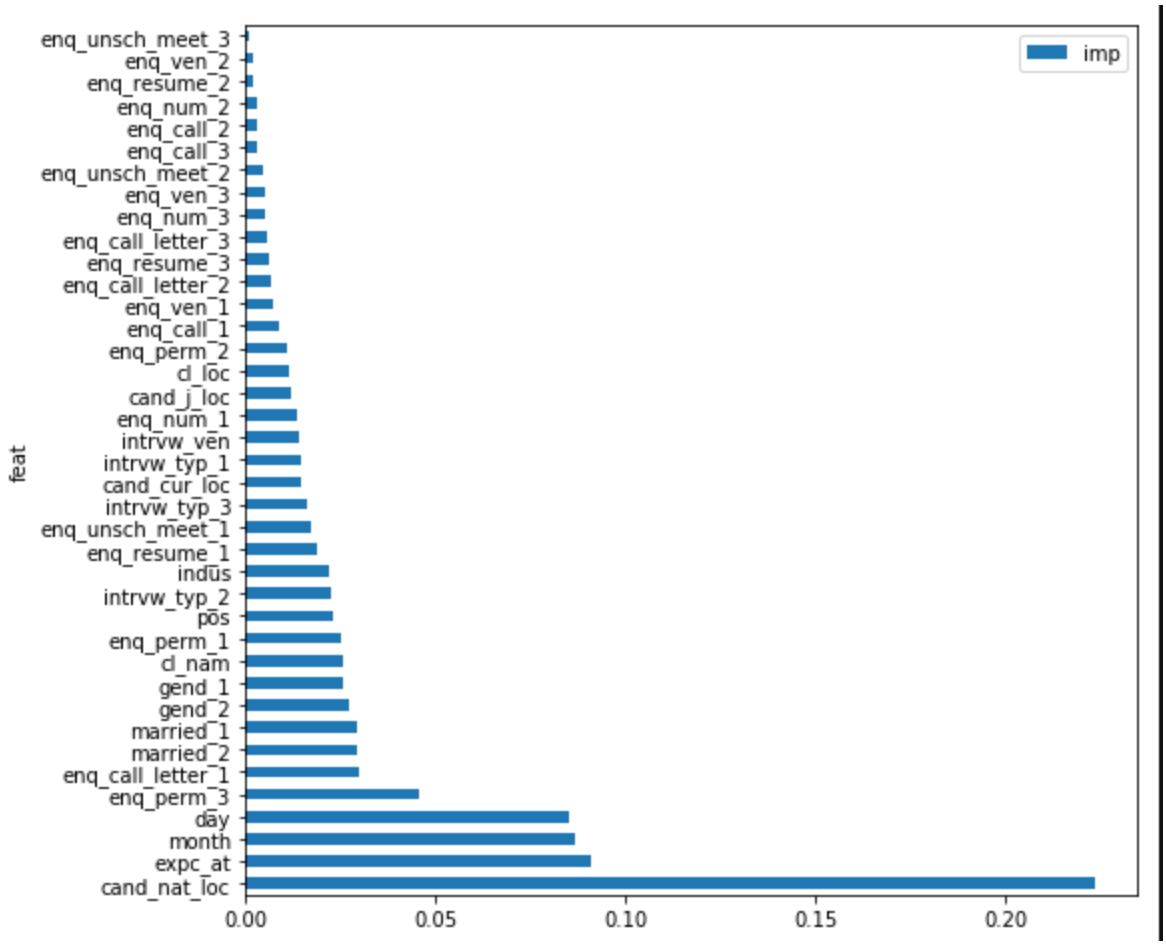
```
x=df2.drop("obs_at",axis=1)
#x=df2.drop(["obs_at","cand_j_loc","enq_call","cl_loc"],axis=1)
y=df2.obs_at
Xtrain,xtest,Ytrain,ytest=model_selection.train_test_split(x,y,test_size=.2,random_state=19,stratify=y)
model=ensemble.RandomForestClassifier(n_estimators=100)
model.fit(Xtrain,Ytrain)
predicted=model.predict(xtest)
printresult(predicted,ytest)
## -----
feat_imp=pd.DataFrame({"imp":model.feature_importances_})
feat_imp["feat"]=Xtrain.columns
feat_imp.sort_values(by="imp",ascending=False,inplace=True)
feat_imp.set_index("feat",inplace=True)
feat_imp.plot.barh(figsize=(8,8))

print("\n after half ohe")
```

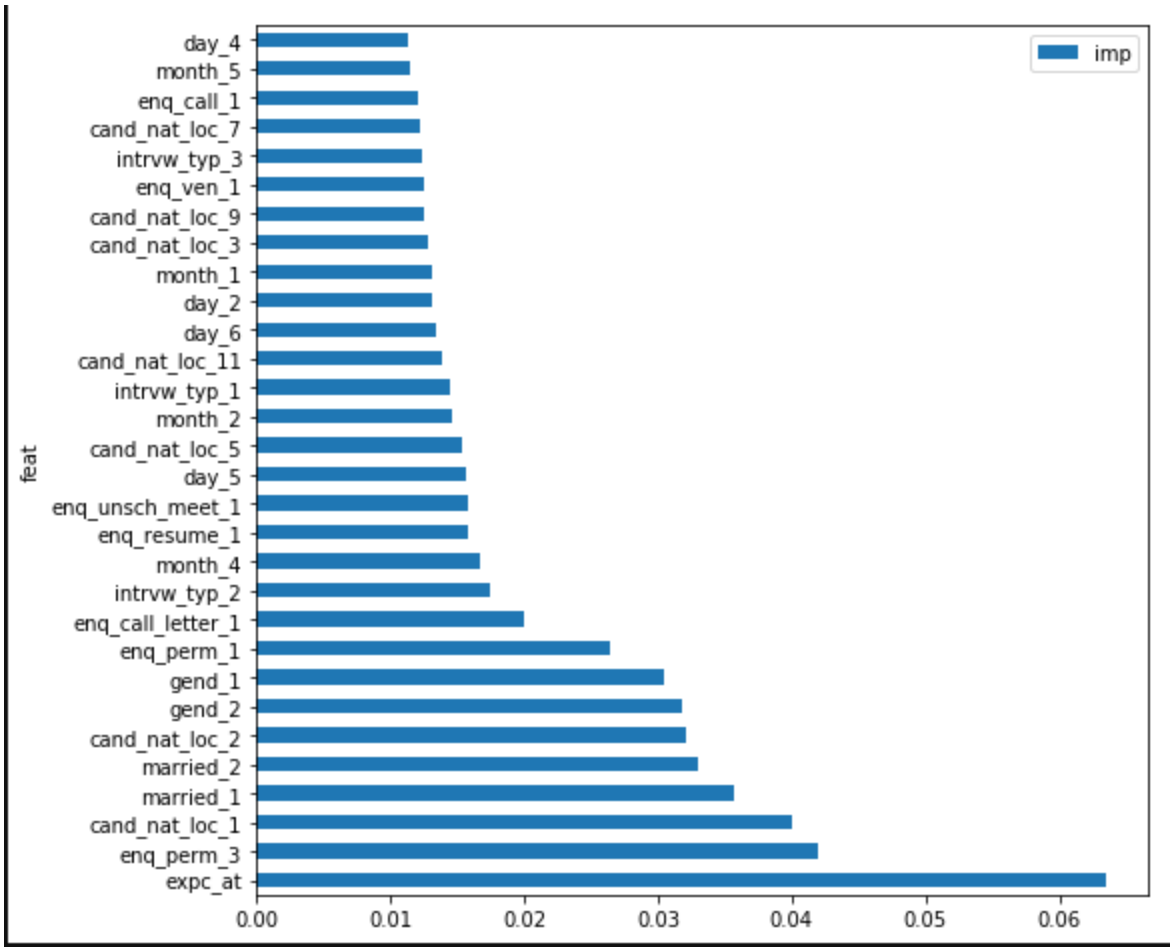
```
x=df4.drop("obs_at",axis=1)
y=df4.obs_at
Xtrain,xtest,Ytrain,ytest=model_selection.train_test_split(x,y,test_size=.2,random_state=42,stratify=y)
model=ensemble.RandomForestClassifier(n_estimators=100)
model.fit(Xtrain,Ytrain)
predicted=model.predict(xtest)
printresult(predicted,ytest)
## -----
feat_imp=pd.DataFrame({"imp":model.feature_importances_})
feat_imp["feat"]=Xtrain.columns
feat_imp.sort_values(by="imp",ascending=False,inplace=True)
feat_imp=feat_imp.nlargest(30,"imp")
feat_imp.set_index("feat",inplace=True)
feat_imp.plot.barh(figsize=(8,8))
```

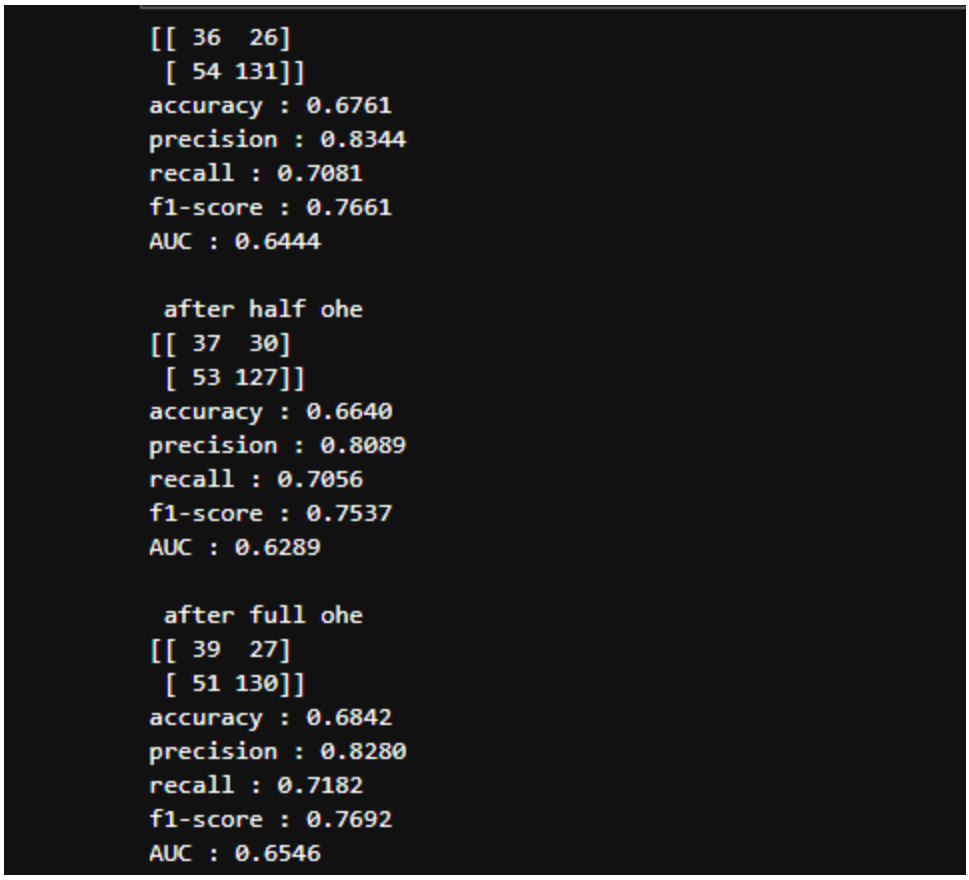
```
[[ 36  26]
 [ 54 131]]
accuracy : 0.6761
precision : 0.8344
recall : 0.7081
f1-score : 0.7661
AUC : 0.6444

 after half ohe
[[ 37  30]
 [ 53 127]]
accuracy : 0.6640
precision : 0.8089
recall : 0.7056
f1-score : 0.7537
AUC : 0.6289

 after full ohe
[[ 39  27]
 [ 51 130]]
accuracy : 0.6842
precision : 0.8280
recall : 0.7182
f1-score : 0.7692
AUC : 0.6546
```

```
x=df2.drop("obs_at",axis=1)
y=df2.obs_at

print("before ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
    print(score,scores,scores.mean())




x=df3.drop("obs_at",axis=1)
y=df3.obs_at

print("\nafter half ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
    print(score,scores,scores.mean())


x=df4.drop("obs_at",axis=1)
y=df4.obs_at

print("\nafter full ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
```


imp
feat
cand_j_loc
enq_call
enq_unsch_meet
cl_loc
intrvw_ven
cand_cur_loc
enq_ven
indus
cl_nam
enq_resume
enq_call_letter
pos
enq_num
intrvw_typ
gend
married
enq_perm
day
month
expc_at
cand_nat_loc
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200


imp
feat
enq_unsch_meet_3
enq_ven_2
enq_resume_2
enq_num_2
enq_call_2
enq_call_3
enq_unsch_meet_2
enq_ven_3
enq_num_3
enq_call_letter_3
enq_resume_3
enq_call_letter_2
enq_ven_1
enq_call_1
enq_perm_2
cl_loc
cand_j_loc
enq_num_1
intrvw_ven
intrvw_typ_1
cand_cur_loc
intrvw_typ_3
enq_unsch_meet_1
enq_resume_1
indus
intrvw_typ_2
pos
enq_perm_1
cl_nam
gend_1
gend_2
married_1
married_2
enq_call_letter_1
enq_perm_3
day
month
expc_at
cand_nat_loc
0.00
0.05
0.10
0.15
0.20


imp
feat
day_4
month_5
enq_call_1
cand_nat_loc_7
intrvw_typ_3
enq_ven_1
cand_nat_loc_9
cand_nat_loc_3
month_1
day_2
day_6
cand_nat_loc_11
intrvw_typ_1
month_2
cand_nat_loc_5
day_5
enq_unsch_meet_1
enq_resume_1
month_4
intrvw_typ_2
enq_call_letter_1
enq_perm_1
gend_1
gend_2
cand_nat_loc_2
married_2
married_1
cand_nat_loc_1
enq_perm_3
expc_at
0.00
0.01
0.02
0.03
0.04
0.05
0.06



Here now the precision is 82% and recall is 71 %. Now we will crossvalidate the data to see if it overfits or underfits. Cross-validation, sometimes called rotation estimation, or out-of-sample testing is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (called the validation dataset or testing set).The goal of cross-validation is to test the

model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, in most methods multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (e.g. averaged) over the rounds to give an estimate of the model's predictive performance.

In summary, cross-validation combines (averages) measures of fitness in prediction to derive a more accurate estimate of model prediction performance. Here we use 5 fold cross validation i.e 5 times validation will be checked.

```
x=df2.drop("obs_at",axis=1)
y=df2.obs_at

print("before ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
    print(score,scores,scores.mean())




x=df3.drop("obs_at",axis=1)
y=df3.obs_at

print("\nafter half ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
    print(score,scores,scores.mean())


x=df4.drop("obs_at",axis=1)
y=df4.obs_at

print("\nafter full ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
```

```
x=df2[col_df2]
y=df2.obs_at
#model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
d=6
model_rfc=ensemble.RandomForestClassifier(n_estimators=100,max_depth=d)
model_rfc.fit(x,y)
estimator=model_rfc.estimators_[0]

export_graphviz(estimator, out_file='tree2.dot',
                feature_names = x.columns.tolist(),
                class_names = ["1","0"],
                rounded = True, proportion = False,
                precision = 2, filled = True)

call(['dot', '-Tpng', 'tree2.dot', '-o', 'tree2.png', '-Gdpi=600'])

##--------------------

x=df3[col_df3]
y=df3.obs_at
model_rfc.fit(x,y)
estimator=model_rfc.estimators_[0]

export_graphviz(estimator, out_file='tree3.dot',
                feature_names = x.columns.tolist(),
                class_names = ["1","0"],
                rounded = True, proportion = False,
                precision = 2, filled = True)
```

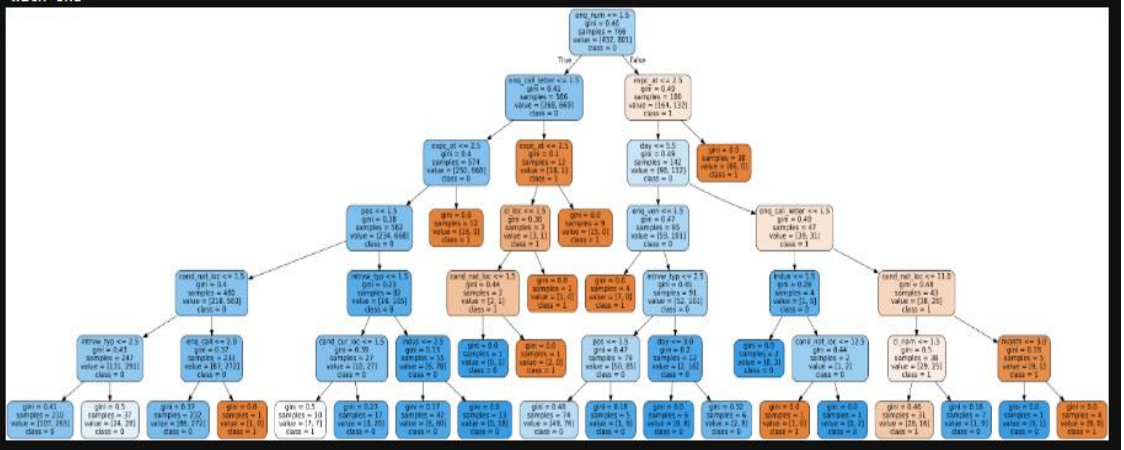
```
call(['dot', '-Tpng', 'tree3.dot', '-o', 'tree3.png', '-Gdpi=600'])

##---------------------
x=df4[col_df4]
y=df4.obs_at
model_rfc.fit(x,y)
estimator=model_rfc.estimators_[0]

export_graphviz(estimator, out_file='tree4.dot',
                feature_names = x.columns.tolist(),
                class_names = ["1","0"],
                rounded = True, proportion = False,
                precision = 2, filled = True)

call(['dot', '-Tpng', 'tree4.dot', '-o', 'tree4.png', '-Gdpi=600'])
```

[224]: 0

[225]:
```
print("without ohe")
Image(filename = 'tree2.png')
```

with ohe

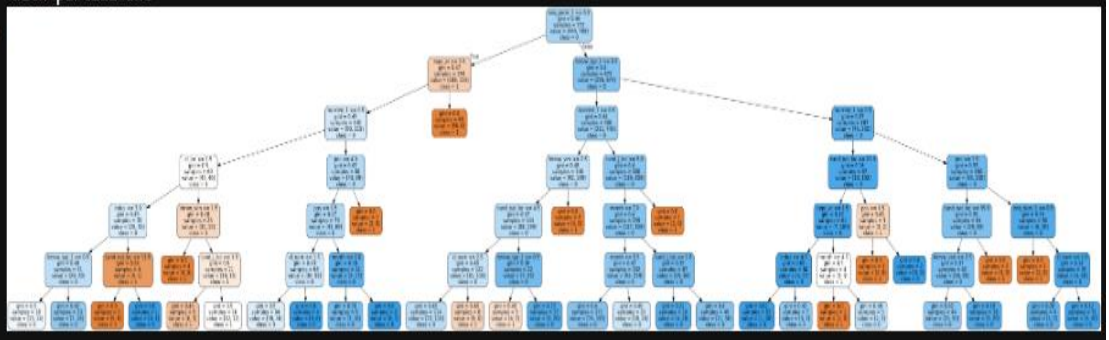
[225]:



[226]:
```
print("with partial ohe")
Image(filename = 'tree3.png')
```

with partial ohe

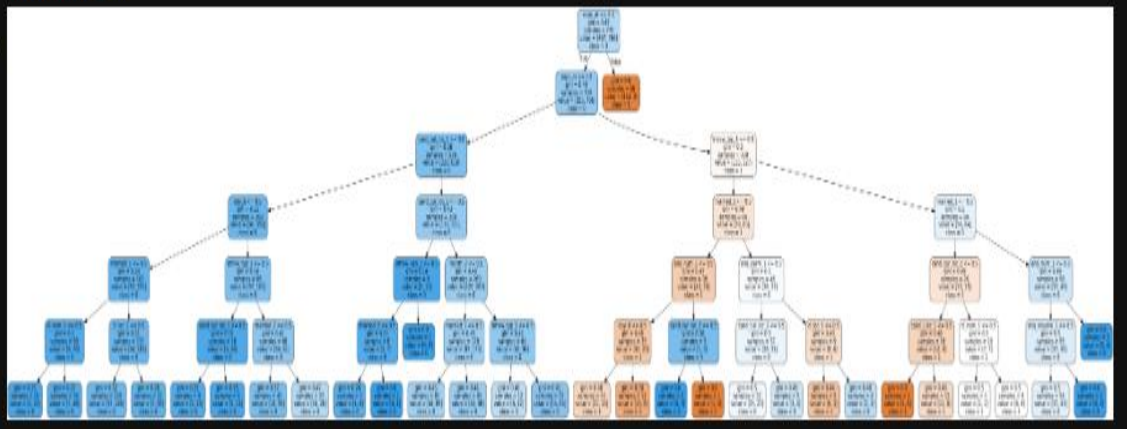
[226]:



[227]:
```
print("with full ohe")
Image(filename = 'tree4.png')
```

with full ohe

[227]:



Results:

Scoring	Scores	Mean	Standard Deviation
accuracy	[0.69635628 0.68016194 0.72064777 0.69105691 0.69512195]	69.67	1.33
precision	[0.67982456 0.77622378 0.75882353 0.705 0.68161435]	72.03	4.0
recall	[0.98726115 0.70063694 0.82165605 0.87820513 0.96794872]	87.11	10.43
f1	[0.80729167 0.74482759 0.79510703 0.78285714 0.80211082]	78.64	2.24
roc_auc	[0.65948337 0.74911536 0.7472753 0.66132479 0.68611111]	70.07	3.99

After cross validation we see here precision is 72.03% and recall is 87%.

Here after cross validation precision decreased and recall improved. Though the precision decreased the model became more generalized so it’s the best model and it overfits.

CONCLUSION

After applying the three models on the cleaned data of the data set we see that random forest is the best method as it gives the highest precision and recall. In random forest method we see after cross validation recall improved and precision decreased so the model overfits.

CODE:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
import calendar
from sklearn import metrics
from sklearn import model_selection
from sklearn import ensemble
from sklearn import linear_model
from sklearn import naive_bayes
from sklearn import feature_selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import VarianceThreshold
from sklearn.tree import export_graphviz
from subprocess import call
from IPython.display import Image
```

```
def printresult(actual,predicted):
    confmatrix=metrics.confusion_matrix(actual,predicted)
    accscore=metrics.accuracy_score(actual,predicted)
    precscore=metrics.precision_score(actual,predicted)
    recscore=metrics.recall_score(actual,predicted)
    print(confmatrix)
    print("accuracy : {:.4f}".format(accscore))
    print("precision : {:.4f}".format(precscore))
    print("recall : {:.4f}".format(recscore))
    print("f1-score : {:.4f}".format(metrics.f1_score(actual,predicted)))
```

```
    print("AUC : {:.4f}".format(metrics.roc_auc_score(actual,predicted)))
```

```
df=pd.read_csv("Interview.csv")
```

```
df.head()
```

	Date of Interview	Client name	Industry	Location	Position to be closed	Nature of Skillset	Interview Type	Name(Cand ID)	Gender	Candidate Current Location	...	Are you clear with the venue details and the landmark.	Has the call letter been shared	Expected Attendance	Observ Attendar
0	13.02.2015	Hospira	Pharmaceuticals	Chennai	Production-Sterile	Routine	Scheduled Walkin	Candidate 1	Male	Chennai	...	Yes	Yes	Yes	
1	13.02.2015	Hospira	Pharmaceuticals	Chennai	Production-Sterile	Routine	Scheduled Walkin	Candidate 2	Male	Chennai	...	Yes	Yes	Yes	
2	13.02.2015	Hospira	Pharmaceuticals	Chennai	Production-Sterile	Routine	Scheduled Walkin	Candidate 3	Male	Chennai	...	NaN	NaN	Uncertain	
3	13.02.2015	Hospira	Pharmaceuticals	Chennai	Production-Sterile	Routine	Scheduled Walkin	Candidate 4	Male	Chennai	...	Yes	Yes	Uncertain	
4	13.02.2015	Hospira	Pharmaceuticals	Chennai	Production-Sterile	Routine	Scheduled Walkin	Candidate 5	Male	Chennai	...	Yes	Yes	Uncertain	

5 rows × 28 columns

e of et	Interview Type	Name(Cand ID)	Gender	Candidate Current Location	...	clear with the venue details and the landmark.	the call letter been shared	Expected Attendance	Observed Attendance	Marital Status	Unnamed: 23	Unnamed: 24	Unnamed: 25	Unnamed: 26	Unnamed: 27
e	Scheduled Walkin	Candidate 1	Male	Chennai	...	Yes	Yes	Yes	No	Single	NaN	NaN	NaN	NaN	NaN
e	Scheduled Walkin	Candidate 2	Male	Chennai	...	Yes	Yes	Yes	No	Single	NaN	NaN	NaN	NaN	NaN
e	Scheduled Walkin	Candidate 3	Male	Chennai	...	NaN	NaN	Uncertain	No	Single	NaN	NaN	NaN	NaN	NaN
e	Scheduled Walkin	Candidate 4	Male	Chennai	...	Yes	Yes	Uncertain	No	Single	NaN	NaN	NaN	NaN	NaN
e	Scheduled Walkin	Candidate 5	Male	Chennai	...	Yes	Yes	Uncertain	No	Married	NaN	NaN	NaN	NaN	NaN

```
print(df.columns[-5:])
df[df.columns[-5:]].describe()
df.drop(df.columns[-5:],axis=1,inplace=True)
```

```
Index(['Unnamed: 23', 'Unnamed: 24', 'Unnamed: 25', 'Unnamed: 26',
       'Unnamed: 27'],
      dtype='object')
```

```
f=open("/home/soham/Desktop/columns","w")
for col in df.columns:
    f.write("\""+col+"\""+" : \"\", \n")
f.close()

f=open("/home/soham/Desktop/markdown","w")
for k in dic.keys():
    f.write("|"+k+"|"+dic[k]+"|\n")
f.close()
```

```
col_dic={"Date of Interview" : "doi",
"Client name" : "cl_nam",
"Industry" : "indus",
"Location" : "cl_loc",
"Position to be closed" : "pos",
"Nature of Skillset" : "skill",
"Interview Type" : "intrvw_typ",
"Name(Cand ID)" : "cand_nam",
"Gender" : "gend",
"Candidate Current Location" : "cand_cur_loc",
"Candidate Job Location" : "cand_j_loc",
"Interview Venue" : "intrvw_ven",
"Candidate Native location" : "cand_nat_loc",
"Have you obtained the necessary permission to start at the required time" : "enq_perm",
"Hope there will be no unscheduled meetings" : "enq_unsch_meet",
"Can I Call you three hours before the interview and follow up on your attendance for the interview" : "enq_call",
"Can I have an alternative number/ desk number. I assure you that I will not trouble you too much" : "enq_num",
"Have you taken a printout of your updated resume. Have you read the JD and understood the same" : "enq_resume",
"Are you clear with the venue details and the landmark." : "enq_ven",
"Has the call letter been shared" : "enq_call_letter",
"Expected Attendance" : "expc_at",
"Observed Attendance" : "obs_at",
"Marital Status" : "married"}
```

```
df.rename(columns=col_dic,inplace=True)
```

Column	Renamed
Date of Interview	doi
Client name	cl_nam
Industry	indus
Location	cl_loc
Position to be closed	pos
Nature of Skillset	skill
Interview Type	intrvw_typ
Name(Cand ID)	cand_nam
Gender	gend
Candidate Current Location	cand_cur_loc
Candidate Job Location	cand_j_loc
Interview Venue	intrvw_ven
Candidate Native location	cand_nat_loc
Have you obtained the necessary permission to start at the required time	enq_perm
Hope there will be no unscheduled meetings	enq_unsch_meet
Can I Call you three hours before the interview and follow up on your attendance for the interview	enq_call
Can I have an alternative number/ desk number. I assure you that I will not trouble you too much	enq_num
Have you taken a printout of your updated resume. Have you read the JD and understood the same	enq_resume

Candidate Job Location	cand_j_loc
Interview Venue	intrvw_ven
Candidate Native location	cand_nat_loc
Have you obtained the necessary permission to start at the required time	enq_perm
Hope there will be no unscheduled meetings	enq_unsch_meet
Can I Call you three hours before the interview and follow up on your attendance for the interview	enq_call
Can I have an alternative number/ desk number. I assure you that I will not trouble you too much	enq_num
Have you taken a printout of your updated resume. Have you read the JD and understood the same	enq_resume
Are you clear with the venue details and the landmark.	enq_ven
Has the call letter been shared	enq_call_letter
Expected Attendance	expc_at
Observed Attendance	obs_at
Marital Status	married

Client Name :

```
]: df1.cl_nam.value_counts()
```

```
]: Standard Chartered Bank           904
   Pfizer                             75
   Hospira                            75
   Aon Hewitt                         28
   Flextronics                        23
   ANZ                                22
   Hewitt                             20
   UST                                18
   Prodapt                            17
   Standard Chartered Bank Chennai    17
   Astrazeneca                        15
   Williams Lea                       11
   Barclays                            5
   Aon hewitt Gurgaon                  2
                                       1
   Woori Bank                          1
   Name: cl_nam, dtype: int64
```

```
l1=[
    "Standard Chartered Bank Chennai",
    "Hewitt",
    "Aon hewitt Gurgaon"
]

l2=[
    "standard chartered bank",
    "aon hewitt",
    "aon hewitt",
]
```

```
df1.cl_nam.replace(l1,l2,inplace=True)
df1.cl_nam=df1.cl_nam.str.lower()
df1.cl_nam.value_counts()
```

```
standard chartered bank    921
pfizer                      75
hospira                     75
aon hewitt                  50
flextronics                 23
anz                         22
ust                         18
prodapt                     17
astrazeneca                 15
williams lea                11
barclays                     5
                            1
woori bank                   1
Name: cl_nam, dtype: int64
```

```
df_temp=df1.copy()
```

Industry :

```
df1.indus.value_counts()
```

```
BFSI                        949
Pharmaceuticals             165
IT Products and Services     45
IT Services                  23
Electronics                  23
Telecom                      17
IT                           11
Name: indus, dtype: int64
```

```
df1.indus.replace(["IT Products and Services","IT Services"],["IT","IT"],inplace=True)
df1.indus=df1.indus.str.lower()
print(df1.indus.value_counts())
```

```
bfsi              949
pharmaceuticals   165
it                 79
electronics        23
telecom            17
Name: indus, dtype: int64
```

```
df_temp=df1.copy()
```

Client Location:

```
print(df1.cl_loc.value_counts())
```

```
Chennai       754
Bangalore     292
chennai        86
Hyderabad      38
Gurgaon        33
Noida          15
- Cochin-       9
chennai          3
Gurgaonr         1
CHENNAI          1
Delhi            1
Name: cl_loc, dtype: int64
```

```
df1.cl_loc=df1.cl_loc.str.lower().str.strip()
df1.cl_loc.replace(["gurgaonr","- cochin-"],["gurgaon","cochin"],inplace=True)
print(df1.cl_loc.value_counts())
```

```
chennai      844
bangalore    292
hyderabad     38
gurgaon       34
noida         15
cochin         9
delhi          1
Name: cl_loc, dtype: int64
```

```
df_temp=df1.copy()
```

Position to be closed:

```
print(df1.pos.value_counts())
```

```
Routine                1023
Niche                   163
Dot Net                  18
Trade Finance            11
AML                       8
Production- Sterile       5
Selenium testing          5
Name: pos, dtype: int64
```

```
df1.pos=df1.pos.str.lower()
print(df1.pos.value_counts())
```

```
routine                1023
niche                   163
dot net                  18
trade finance            11
aml                       8
production- sterile       5
selenium testing          5
Name: pos, dtype: int64
```

Nature of skill set:

too many weird values. Will look at it later

```
print(df1.skill.value_counts()[:5])
```

```
JAVA/J2EE/Struts/Hibernate    220
Fresher                        86
Accounting Operations          86
AML/KYC/CDD                    84
CDD KYC                        52
Name: skill, dtype: int64
```

Interview type:

```
df1.intrvw_typ.value_counts()
```

```
Scheduled Walk In    456
Scheduled            371
Walkin               189
Scheduled Walkin     189
Walkin                27
Sceduled walkin        1
Name: intrvw_typ, dtype: int64
```

```
df1.intrvw_typ=df1.intrvw_typ.str.lower().str.strip()
df1.intrvw_typ.replace(["scheduled walk in","sceduled walkin"],["scheduled walkin","scheduled walkin"],inplace=True)
df1.intrvw_typ.value_counts()
```

```
scheduled walkin    646
scheduled           371
walkin              216
Name: intrvw_typ, dtype: int64
```

Gender:

```
df1.gend=df1.gend.str.lower().str.strip()
```

Candidate current location:

```
df1.cand_cur_loc.value_counts()
```

```
Chennai       754
Bangalore     292
chennai        86
Hyderabad      38
Gurgaon        34
Noida          15
- Cochin-       9
chennai         3
CHENNAI         1
Delhi           1
Name: cand_cur_loc, dtype: int64
```

```
df1.cand_cur_loc=df1.cand_cur_loc.str.lower().str.strip()
df1.cand_cur_loc.replace(["- cochin-"],["cochin"],inplace=True)
print(df1.cand_cur_loc.value_counts())
```

```
chennai      844
bangalore    292
hyderabad     38
gurgaon       34
noida         15
cochin         9
delhi          1
Name: cand_cur_loc, dtype: int64
```

Interview venue:

```
df1.intrvw_ven.value_counts()
```

```
Chennai       852
Bangalore     277
Hyderabad      40
Gurgaon        35
Noida          15
- Cochin-       9
Hosur           5
Name: intrvw_ven, dtype: int64
```

```
df1.intrvw_ven=df1.intrvw_ven.str.lower().str.strip()
df1.intrvw_ven.replace(["- cochin-"],["cochin"],inplace=True)
print(df1.intrvw_ven.value_counts())
```

```
chennai      852
bangalore    277
hyderabad     40
gurgaon       35
noida         15
cochin         9
hosur          5
Name: intrvw_ven, dtype: int64
```

```
df_temp=df1.copy()
```

Candidate native location:

dont know what to do with this

```
l1=df1.cand_nat_loc.str.lower().str.strip().value_counts().index.tolist()
```

Have you obtained the necessary permission to start at the required time

```
df1.enq_perm.value_counts()
```

```
Yes               917
No                 79
Not yet            19
Na                  5
yes                 4
Yet to confirm      4
NO                  1
Name: enq_perm, dtype: int64
```

```
df1.enq_perm=df1.enq_perm.str.lower()
df1.enq_perm.replace(["not yet","na","yet to confirm"],["no","no","no"],inplace=True)
print(df1.enq_perm.value_counts())
```

```
yes    921
no     108
Name: enq_perm, dtype: int64
```

```
#df_temp=df1.copy
df1.enq_perm.isna().sum()
```

```
205
```

```
df1['enq_perm'].fillna("uncertain",inplace=True)
df1.drop(df.index[[1233]], inplace = True)
```

Hope there will be no unscheduled meetings

```
df1.enq_unsch_meet.str.lower().value_counts()
```

```
yes         954
na           20
no            6
not sure      5
cant say      1
Name: enq_unsch_meet, dtype: int64
```

```
df1.enq_unsch_meet=df1.enq_unsch_meet.str.lower()
df1.enq_unsch_meet.replace(["na","cant say","not sure"],["no","uncertain","uncertain"],inplace=True)
df1.enq_unsch_meet.value_counts()
```

```
yes          954
no            26
uncertain      6
Name: enq_unsch_meet, dtype: int64
```

```
df1.enq_unsch_meet.isna().sum()
```

```
247
```

```
df1.enq_unsch_meet.fillna("uncertain",inplace=True)
```

```
df_temp=df1.copy()
```

Can I Call you three hours before the interview and follow up on your attendance for the interview

```
df1.enq_call.value_counts()
```

```
Yes        951
Na          20
No          10
yes          4
No Dont      1
Name: enq_call, dtype: int64
```

```
df1.enq_call=df1.enq_call.str.lower()
df1.enq_call.replace(["na","no dont"],["no","no"],inplace=True)
df1.enq_call.value_counts()
```

```
yes    955
no      31
Name: enq_call, dtype: int64
```

```
df1.enq_call=df1.enq_call.str.lower()
df1.enq_call.replace(["na","no dont"],["no","no"],inplace=True)
df1.enq_call.value_counts()
```

```
yes    955
no      31
Name: enq_call, dtype: int64
```

```
df1.enq_call.fillna("uncertain",inplace=True)
```

```
df_temp=df1.copy()
```

Can I have an alternative number/ desk number. I assure you that I will not trouble you too much

```
df1.enq_num.value_counts()
```

```
Yes                          936
No                            27
Na                            19
No I have only thi number      2
yes                            1
na                             1
Name: enq_num, dtype: int64
```

```
df1.enq_num=df1.enq_num.str.lower()
df1.enq_num.replace(["na","no i have only thi number"],["no","no"],inplace=True)
print(df1.enq_num.value_counts())
```

```
yes    937
no      49
Name: enq_num, dtype: int64
```

```
df1.enq_num.fillna("uncertain",inplace=True)
```

Have you taken a printout of your updated resume. Have you read the JD and understood the same

```
df1.enq_resume.value_counts()
```

```
Yes                      940
Na                        19
No                        16
Not Yet                    4
Not yet                    2
yes                        2
No- will take it soon      1
na                         1
Name: enq_resume, dtype: int64
```

```
df1.enq_resume=df1.enq_resume.str.lower()
df1.enq_resume.replace(["na","not yet","no- will take it soon"],["no","no","no"],inplace=True)
df1.enq_resume.value_counts()
```

```
yes    942
no      43
Name: enq_resume, dtype: int64
```

```
df1.enq_resume.fillna("uncertain",inplace=True)
```

```
df_temp=df1.copy()
```

Are you clear with the venue details and the landmark.

```
df1.enq_ven.value_counts()
```

```
Yes                    946
Na                      19
No                      14
yes                      2
No- I need to check      2
na                       1
no                       1
Name: enq_ven, dtype: int64
```

```
df1.enq_ven=df1.enq_ven.str.lower()
df1.enq_ven.replace(["na","no- i need to check"],["no","no"],inplace=True)
print(df1.enq_ven.value_counts())
```

```
yes    948
no      37
Name: enq_ven, dtype: int64
```

```
df1.enq_ven.fillna("uncertain",inplace=True)
```

```
df_temp=df1.copy()
```

Has the call letter been shared

```
df1.enq_call_letter.value_counts()
```

```
Yes               932
Na                 19
No                 17
Not Sure            8
Need To Check       3
Not yet             2
yes                 2
Havent Checked      1
na                  1
Not sure            1
Yet to Check        1
no                  1
Name: enq_call_letter, dtype: int64
```

```
df1.enq_call_letter=df1.enq_call_letter.str.lower()
df1.enq_call_letter.replace(["na","not yet","not sure","havent checked","yet to check","need to check"],["no","no","uncertain","u
print(df1.enq_call_letter.value_counts())
```

```
yes         934
no           40
uncertain    14
Name: enq_call_letter, dtype: int64
```

```
df1.enq_call_letter.fillna("uncertain",inplace=True)
df1.enq_call_letter.value_counts()
```

```
yes         934
uncertain   259
no           40
Name: enq_call_letter, dtype: int64
```

```
df_temp=df1.copy()
```

Expected Attendance

```
df1.expc_at.value_counts()
```

```
Yes          882
Uncertain    250
No            59
NO            34
yes            1
10.30 Am       1
11:00 AM       1
Name: expc_at, dtype: int64
```

```
df1.expc_at=df1.expc_at.str.lower()
df1.expc_at.replace(["11:00 am","10.30 am"],["yes","yes"],inplace=True)
print(df1.expc_at.value_counts())
```

```
yes          885
uncertain    250
no            93
Name: expc_at, dtype: int64
```

```
df1.expc_at.fillna("uncertain",inplace=True)
```

```
df_temp=df1.copy()
```

Observed Attendance

```
df1.obs_at.value_counts()
```

```
Yes     701
No      401
yes      81
NO       35
no        7
No        6
no        1
yes       1
Name: obs_at, dtype: int64
```

```
df1.obs_at=df1.obs_at.str.lower().str.strip()
print(df1.obs_at.value_counts())
```

```
yes    783
no     450
Name: obs_at, dtype: int64
```

```
df_temp=df1.copy()
```

Marital Status

```
df1.married.value_counts()
```

```
Single     767
Married    466
Name: married, dtype: int64
```

```
df1.married=df1.married.str.lower()
print(df1.married.value_counts())
```

```
single     767
married    466
Name: married, dtype: int64
```

```
df_temp=df1.copy()
```

DATE OF INTERVIEW

```
#This is used to format the dates
def clean_date(date):
    date = date.str.strip()
    date = date.str.split("&").str[0]
    date = date.str.replace('-', '/')
    date = date.str.replace('.', '/')
    date = date.str.replace('Apr', '04')
    date = date.str.replace('-', '/')
    date = date.str.replace(' ', '/')
    date = date.str.replace('//+', '/')
    return date
```

modeling_df = modeling_df[modeling_df['date'] < '2018-01-01']

```
df1['doi'] = clean_date(df1['doi'])
```

```
type(df1.doi[0])
```

```
str
```

```
#To make all the values in the same format of date time to get the day of interview and the month of inter
df1['year'] = df1['doi'].str.split("/").str[2]
df1['day'] = df1['doi'].str.split("/").str[0]
df1['month'] = df1['doi'].str.split("/").str[1]
df1['year'].replace(['16', '15'], ['2016', '2015'], inplace = True)
df1['date'] = pd.to_datetime(pd.DataFrame({'year': df1['year'],
                                           'month': df1['month'],
                                           'day': df1['day']}), format = '%Y-%m-%d')
df1.drop(['year', 'month', 'day'], axis = 1, inplace = True)
```

```
dt=df1['date']
```

```
ls=[]
for i in range(len(dt)):
    ls.append(dt[i].month)
df1['month']=ls
```

```
ls=[]
for i in range(len(dt)):
    ls.append(dt[i].weekday()+1)
df1['day']=ls
```

```
df1.drop(columns=['doi','date','skill'],axis=1,inplace=True)
```

```
df_temp=df1.copy()
```

```
df_temp=df1.copy()
```

```
for col in df1.columns:
    print(col," ",df1[col].isna().sum())
```

```
cl_nam   0
indus   0
cl_loc   0
pos   0
intrvw_typ   0
cand_nam   0
gend   0
cand_cur_loc   0
cand_j_loc   0
intrvw_ven   0
cand_nat_loc   0
enq_perm   0
enq_unsch_meet   0
enq_call   0
enq_num   0
```

Cleaned:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1233 entries, 0 to 1232
Data columns (total 23 columns):
cl_nam              1233 non-null object
indus               1233 non-null object
cl_loc              1233 non-null object
pos                 1233 non-null object
intrvw_typ          1233 non-null object
cand_nam            1233 non-null object
gend                1233 non-null object
cand_cur_loc        1233 non-null object
cand_j_loc          1233 non-null object
intrvw_ven          1233 non-null object
cand_nat_loc        1233 non-null object
```

```
enq_perm            1233 non-null object
enq_unsch_meet      1233 non-null object
enq_call            1233 non-null object
enq_num             1233 non-null object
enq_resume          1233 non-null object
enq_ven             1233 non-null object
enq_call_letter     1233 non-null object
expc_at             1233 non-null object
obs_at              1233 non-null object
married             1233 non-null object
month               1233 non-null int64
day                 1233 non-null int64
dtypes: int64(2), object(21)
memory usage: 271.2+ KB
```

One Hot Encoding:

```
df1.drop('cand_nam',axis=1,inplace=True)
```

```
df2=df1.copy()
def t(col):
    j=1
    dic={}
    for i in df2[col].value_counts().index:
        dic[i]=j
        if col=="obs_at" :
            j-=1
        else:
            j+=1
    df2[col].replace(dic,inplace=True)
```

```
for col in df2.columns:
    if(col=='month' or col=='day'):
        continue
    t(col)
```

```
df3 = df2.copy()
df4=df2.copy()
```

```
ohel_half = ['intrvw_typ','gend','enq_perm','enq_unsch_meet','enq_call','enq_num','enq_resume','enq_ven','enq_call_letter','marr
ohel_full=['cl_nam', 'indus', 'cl_loc', 'pos', 'intrvw_typ', 'gend',
       'cand_cur_loc', 'cand_j_loc', 'intrvw_ven', 'cand_nat_loc', 'enq_perm',
       'enq_unsch_meet', 'enq_call', 'enq_num', 'enq_resume', 'enq_ven',
       'enq_call_letter', 'married', 'month', 'day']
```

```
ohel_half = ['intrvw_typ','gend','enq_perm','enq_unsch_meet','enq_call','enq_num','enq_resume','enq_ven','enq_call_letter','m
ohel_full=['cl_nam', 'indus', 'cl_loc', 'pos', 'intrvw_typ', 'gend',
       'cand_cur_loc', 'cand_j_loc', 'intrvw_ven', 'cand_nat_loc', 'enq_perm',
       'enq_unsch_meet', 'enq_call', 'enq_num', 'enq_resume', 'enq_ven',
       'enq_call_letter', 'married', 'month', 'day']
df3 = pd.get_dummies(df3, prefix=ohel_half, columns=ohel_half)
df4 = pd.get_dummies(df4,prefix=ohel_full,columns=ohel_full)
```

```
print(len(df3.columns))
print(len(df4.columns))
```

```
40
145
```

Random Forest Classifier:

Calculating a baseline

draw the roc curve and tree graph

```
x=df2.drop("obs_at",axis=1)
#x=df2.drop(["obs_at","cand_j_loc","enq_call","cl_loc"],axis=1)
y=df2.obs_at
Xtrain,xtest,Ytrain,ytest=model_selection.train_test_split(x,y,test_size=.2,random_state=19,stratify=y)
model=ensemble.RandomForestClassifier(n_estimators=100)
model.fit(Xtrain,Ytrain)
predicted=model.predict(xtest)
printresult(predicted,ytest)
## -----
feat_imp=pd.DataFrame({"imp":model.feature_importances_})
feat_imp["feat"]=Xtrain.columns
feat_imp.sort_values(by="imp",ascending=False,inplace=True)
feat_imp.set_index("feat",inplace=True)
```

```
print("\n after half ohe")

x=df3.drop("obs_at",axis=1)
y=df3.obs_at
Xtrain,xtest,Ytrain,ytest=model_selection.train_test_split(x,y,test_size=.2,random_state=42,stratify=y)
model=ensemble.RandomForestClassifier(n_estimators=100)
model.fit(Xtrain,Ytrain)
predicted=model.predict(xtest)
printresult(predicted,ytest)
## -----
feat_imp=pd.DataFrame({"imp":model.feature_importances_})
feat_imp["feat"]=Xtrain.columns
feat_imp.sort_values(by="imp",ascending=False,inplace=True)
feat_imp.set_index("feat",inplace=True)
feat_imp.plot.barh(figsize=(8,8))
```

```
print("\n after full ohe")

x=df4.drop("obs_at",axis=1)
y=df4.obs_at
Xtrain,xtest,Ytrain,ytest=model_selection.train_test_split(x,y,test_size=.2,random_state=42,stratify=y)
model=ensemble.RandomForestClassifier(n_estimators=100)
model.fit(Xtrain,Ytrain)
predicted=model.predict(xtest)
printresult(predicted,ytest)
## -----
feat_imp=pd.DataFrame({"imp":model.feature_importances_})
feat_imp["feat"]=Xtrain.columns
feat_imp.sort_values(by="imp",ascending=False,inplace=True)
feat_imp=feat_imp.nlargest(30,"imp")
feat_imp.set_index("feat",inplace=True)
feat_imp.plot.barh(figsize=(8,8))
```

Cross validating:

5 fold

```
# for score in ["accuracy", "precision", "recall"]:
x=df2.drop("obs_at",axis=1)
y=df2.obs_at

print("before ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
    print(score,scores,scores.mean())



x=df3.drop("obs_at",axis=1)
y=df3.obs_at

print("\nafter half ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
    print(score,scores,scores.mean())
```

```
x=df4.drop("obs_at",axis=1)
y=df4.obs_at

print("\nafter full ohe")
model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    scores=model_selection.cross_val_score(model_rfc,x,y,cv=5,scoring=score)
    print(score,scores,scores.mean())
```

Feature selection:

KBest:

```
print("\nwithout ohe")
x=df2.drop("obs_at",axis=1)
y=df2.obs_at
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(8,'Score'))  #print 8 best features
col_df2=featureScores.nlargest(8,"Score")["Specs"].tolist()
```

```
print("\n\nafter half ohe")
x=df3.drop("obs_at",axis=1)
y=df3.obs_at
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(8,'Score'))  #print 8 best features
col_df3=featureScores.nlargest(8,"Score")["Specs"].tolist()

print("\n\nafter full ohe")
x=df4.drop("obs_at",axis=1)
y=df4.obs_at
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
```

```
x=df2[col_df2]
y=df2.obs_at
#model_rfc=ensemble.RandomForestClassifier(n_estimators=100)
d=6
model_rfc=ensemble.RandomForestClassifier(n_estimators=100,max_depth=d)
model_rfc.fit(x,y)
estimator=model_rfc.estimators_[0]

export_graphviz(estimator, out_file='tree2.dot',
                feature_names = x.columns.tolist(),
                class_names = ["1","0"],
                rounded = True, proportion = False,
                precision = 2, filled = True)

call(['dot', '-Tpng', 'tree2.dot', '-o', 'tree2.png', '-Gdpi=600'])
```

```
x=df4[col_df4]
y=df4.obs_at
model_rfc.fit(x,y)
estimator=model_rfc.estimators_[0]

export_graphviz(estimator, out_file='tree4.dot',
                feature_names = x.columns.tolist(),
                class_names = ["1","0"],
                rounded = True, proportion = False,
                precision = 2, filled = True)

call(['dot', '-Tpng', 'tree4.dot', '-o', 'tree4.png', '-Gdpi=600'])
```

Logistic Regression

Feature selection:

```
X = df3.drop(['obs_at','expc_at'],axis=1)
y = df3['obs_at']
thresholder = VarianceThreshold(threshold=.22)
thresholder.fit_transform(X)
temp = X.columns[thresholder.get_support()]
temp
```

```
Index(['cl_nam', 'indus', 'cl_loc', 'pos', 'cand_cur_loc', 'cand_j_loc',
       'intrvw_ven', 'cand_nat_loc', 'month', 'day', 'intrvw_typ_1',
       'married_1', 'married_2'],
      dtype='object')
```

```
feature_scores = feature_selection.mutual_info_classif(X, y)
micc = []
for score, fname in sorted(zip(feature_scores, X.columns.tolist()), reverse=True)[:10]:
    micc += [fname]
micc
```

```
['enq_perm_3',
 'enq_call_letter_1',
 'enq_unsch_meet_2',
 'enq_call_1',
 'enq_unsch_meet_1',
 'enq_call_letter_2',
 'enq_perm_1',
 'enq_num_1',
```

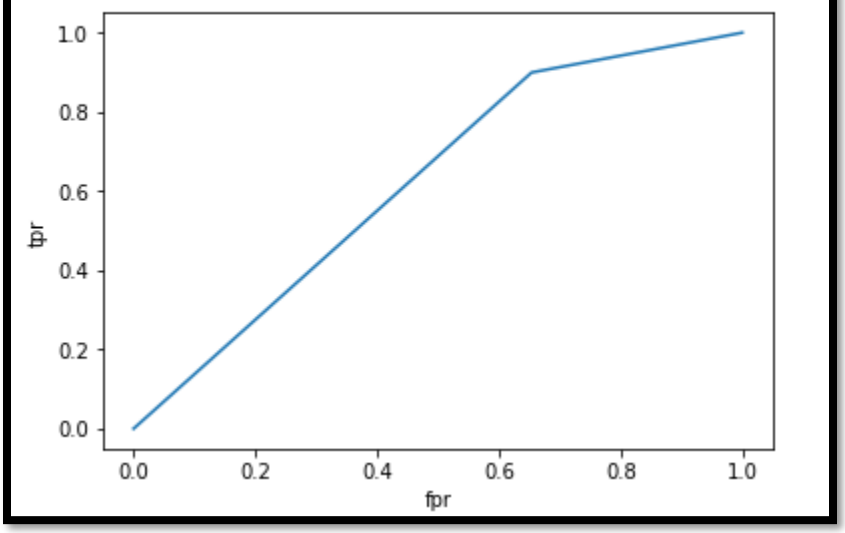
Naive Bayes

```
model=naive_bayes.BernoulliNB()
model.fit(Xtrain,ytrain)
predict_test=model.predict(Xtest)
printresult(ytest,predict_test)
```

```
[[ 27  51]
 [ 17 152]]
accuracy : 0.7247
precision : 0.7488
recall : 0.8994
f1-score : 0.8172
AUC : 0.6228
```

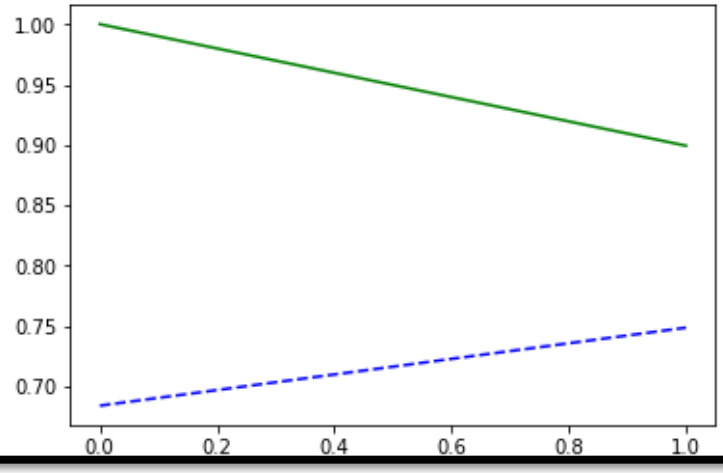
```
tpr,fpr,threshhold=metrics.roc_curve(ytest,predict_test)
plt.plot(tpr,fpr)
plt.xlabel("fpr")
plt.ylabel("tpr")
```

```
Text(0, 0.5, 'tpr')
```

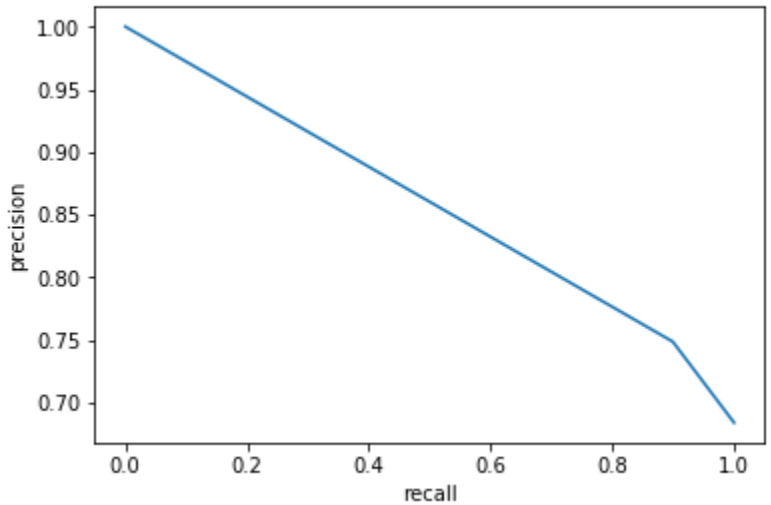


```
prec,recall,threshhold=metrics.precision_recall_curve(ytest,predict_test)
plt.plot(threshhold,prec[:-1],'b--')
plt.plot(threshhold,recall[:-1],'g-')
```

```
[<matplotlib.lines.Line2D at 0x7f46ac241128>]
```




```
Text(0, 0.5, 'precision')
```



```
tmodel=0
model=naive_bayes.BernoulliNB()
for score in ["accuracy", "precision", "recall","f1","roc_auc"]:
    modelcvscore=model_selection.cross_val_score(model,X,y,cv=5,scoring=score)
    if score=="f1":
        print("f1-score",":",round(modelcvscore.mean()*100))
    elif score=="roc_auc":
        print("AUC",":",round(modelcvscore.mean()*100))
    else:
        print(score,":",round(modelcvscore.mean()*100))
```

```
accuracy : 70.0
precision : 72.0
recall : 86.0
f1-score : 78.0
AUC : 64.0
```

Results:

Scoring	Scores
accuracy	70.0
precision	72.0
recall	86.0
f1-score	78.0
AUC	64.0

FUTURE SCOPE OF IMPROVEMENTS

We have gathered the analysis of data from the data sheet, it can be possible furthermore to predict future. If the information have been more precise then the future predictions would have been more accurate.

CERTIFICATE

This is to certify that Ms. CHANDRABALI BISHNU of RCCIIT,
Registration number: 161170110026, has successfully completed a project on INTERVIEW PREDICTION using MACHINE LEARNING and PYTHON under the guidance of Mr. TITAS ROYCHOWDHURY.

[*Name of your faculty*]

Globsyn Finishing School

CERTIFICATE

This is to certify that Ms. CAMELIA MAHATO of RCCIIT,
Registration number: 161170110025, has successfully completed a project on INTERVIEW PREDICTION using MACHINE LEARNING and PYTHON under the guidance of Mr. TITAS ROYCHOWDHURY.

[*Name of your faculty*]

Globsyn Finishing School

CERTIFICATE

This is to certify that Mr. PALLAB CHAKRABORTY of RCCIIT,
Registration number: 161170110048, has successfully completed a project ON INTERVIEW PREDICTION using MACHINE LEARNING and PYTHON under the guidance of Mr. TITAS ROYCHOWDHURY.

[*Name of your faculty*]

Globsyn Finishing School

CERTIFICATE

This is to certify that Mr. SOHAM MANDAL of RCCIIT,
Registration number: 161170110070, has successfully completed a project on INTERVIEW PREDICTION using MACHINE LEARNING and PYTHON under the guidance of Mr. TITAS ROYCHOWDHURY.

[*Name of your faculty*]

Globsyn Finishing School

CERTIFICATE

This is to certify that Mr. ANIKET CHATTOPADHYAY of RCCIIT, Registration number: 161170110007, has successfully completed a project on INTERVIEW PREDICTION using MACHINE LEARNING and PYTHON under the guidance of Mr. TITAS ROYCHOWDHURY.

[*Name of your faculty*]

Globsyn Finishing School