1) Data Integeration

In [1]:
```
import json
import ast
import pandas as pd
```

read json file and flatten list

In [2]:
```
df = pd.read_json('c:/Users/Rozita/Documents/jo app 2021/tempus_data_science_case
```

In [3]:
```
df #it has nested list
```

Out[3]:

	institution	cohort_id	patient_profiles
0	Saint Penelope Medical Center	14562556998	[{'patient_id': '102bb8fae', 'demographics': {...
1	BioLab, Inc.	14562556998	[{'patient_id': '100688fb9', 'demographics': {...
2	University Hospital System	14562556998	[{'patient_id': '1002df1d3', 'demographics': {...
3	Goodfellow Research Institute	14562556998	[{'patient_id': '104fc5e3c', 'demographics': {...
4	Montague Hospital	14562556998	[{'patient_id': '1010441f', 'demographics': {'...
5	Johnson & Bloom Hospitals	14562556998	[{'patient_id': '103278b88', 'demographics': {...
6	Medical Information Exchange	14562556998	[{'patient_id': '1002cb1e8', 'demographics': {...

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   institution       7 non-null      object
 1   cohort_id         7 non-null      int64 
 2   patient_profiles  7 non-null      object
dtypes: int64(1), object(2)
memory usage: 296.0+ bytes
```

In [5]:
```
# Load data using Python JSON module
with open('C:/Users/Rozita/Documents/jo app 2021/tempus_data_science_case_study/t
    data = json.loads(f.read())
```

In [6]:
```
data
```

```
'status': {'disease_sub_type': 'A',
  'comorbidity_index': None,
  'cohort_qualifier': True,
  'smoking_status': 'never',
  'months_since_diagnosis': 0}},
 {'patient_id': '11b1d32a1',
  'demographics': {'gender': 'Male', 'age': 47, 'race': 'White'},
  'status': {'disease_sub_type': 'A',

  'comorbidity_index': None,
  'cohort_qualifier': True,
  'smoking_status': 'never',
  'months_since_diagnosis': 0}},
 {'patient_id': '11c6138f1',
  'demographics': {'gender': 'Female',
  'age': 65,
  'race': 'Black or African American'},
  'status': {'disease_sub_type': 'A',
  'comorbidity_index': 2,
  'cohort_qualifier': True,
```

In [7]:
```
# Flatten data
df_nested_list = pd.json_normalize(data, record_path =['patient_profiles'])
```

In [8]:
```
df_nested_list
```

Out[8]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.comorbic
0	102bb8fae	Female	68.0	A	
1	10e32947f	Female	66.0	A	
2	11156e14a	Male	61.0	A	
3	113d8066d	Male	62.0	B	
4	113ec3f1	Male	59.0	A	
...	...	...	...	...	
2346	f9535f0b	Female	70.0	A	
2347	fa0fa338	Male	77.0	A	
2348	fdb519f4	Female	68.0	A	
2349	fdd3b1c7	Male	56.0	F	
2350	ff9f0a0	Female	56.0	A	

2351 rows × 13 columns

In [9]:
```
# To include 'institution', 'cohort_id
df_nested_list_pt_profile = pd.json_normalize(
    data,
    record_path =['patient_profiles'],
    meta=['institution', 'cohort_id']
)
```

In [10]:
```
df_nested_list_pt_profile
```

Out[10]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.comorbic
0	102bb8fae	Female	68.0	A	
1	10e32947f	Female	66.0	A	
2	11156e14a	Male	61.0	A	
3	113d8066d	Male	62.0	B	
4	113ec3f1	Male	59.0	A	
...	...	...	...	...	
2346	f9535f0b	Female	70.0	A	
2347	fa0fa338	Male	77.0	A	
2348	fdb519f4	Female	68.0	A	
2349	fdd3b1c7	Male	56.0	F	
2350	ff9f0a0	Female	56.0	A	

2351 rows × 15 columns

read biomarkers csv file

In [12]:
```
dfbiomarkers = pd.read_csv ('c:/Users/Rozita/Documents/jo app 2021/tempus_data_sc
dfbiomarkers
```

Out[12]:

	biomarker_id	BM00000	BM00001	BM00002	BM00003	BM00004	BM00005	BM00006	BM00
0	100505de2	1	1	1	0	0	0	1	
1	10075c5c2	1	1	1	0	0	0	1	
2	10105cb22	1	1	1	0	0	0	1	
3	101219d6e	1	0	1	0	0	0	0	
4	10135ad45	1	0	1	0	1	0	1	
...	...	...	...	...	...	...	...	...	
1955	feaf2df	1	1	1	0	0	0	1	
1956	ff65d86c	1	1	1	0	1	0	0	
1957	ff9361bb	1	1	1	0	1	0	1	
1958	ffaae762	0	1	1	0	1	1	1	
1959	ffdfe7f0	1	1	1	0	1	0	1	

1960 rows × 15158 columns

read Targets file

In [13]:
```
dftargets = pd.read_csv ('c:/Users/Rozita/Documents/jo app 2021/tempus_data_scier
dftargets
```

Out[13]:

	patient_id	biomarker_id	target_label
0	1002df1d3	89c43bb4	0
1	1010441f	3ae31327	0
2	101eb6af1	1d8f7bab9	0
3	10204394e	129ee8c6c	0
4	1021d329b	1b44145bd	0
...	...	...	...
1729	f935591e	5f456337	1
1730	fa4be3df	1b4e0b08f	1
1731	fd416b4b	c8c12c4a	1
1732	fdaeb5f1	222e21be2	1
1733	ffcfa280	185548f2f	1

1734 rows × 3 columns

merge dataframes targets and biomarkers

In [14]:
```
# Merge DataFrames by Columns
df3_target_biomarker=pd.merge(dfbiomarkers,dftargets, on='biomarker_id')
df3_target_biomarker
```

Out[14]:

	biomarker_id	BM00000	BM00001	BM00002	BM00003	BM00004	BM00005	BM00006	BM00
0	100505de2	1	1	1	0	0	0	1	
1	10075c5c2	1	1	1	0	0	0	1	
2	10105cb22	1	1	1	0	0	0	1	
3	101219d6e	1	0	1	0	0	0	0	
4	10135ad45	1	0	1	0	1	0	1	
...	...	...	...	...	...	...	...	...	
1729	fe62a76e	1	1	1	0	0	0	0	
1730	ff65d86c	1	1	1	0	1	0	0	
1731	ff9361bb	1	1	1	0	1	0	1	
1732	ffaae762	0	1	1	0	1	1	1	
1733	ffdfe7f0	1	1	1	0	1	0	1	

1734 rows × 15160 columns

In [15]:
```
# 5 merge dataframes df3_target_biomarker and json file on patient_id
```

In [16]:
```
# Merge DataFrames by Columns
df4_target_biomarker_json=pd.merge(df_nested_list_pt_profile, df3_target_biomarke
df4_target_biomarker_json
```

Out[16]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.comorbic
0	11156e14a	Male	61.0	A	
1	113d8066d	Male	62.0	B	
2	113ec3f1	Male	59.0	A	
3	114a37875	Male	68.0	B	
4	11b1d32a1	Male	47.0	A	
...	...	...	...	...	
1729	d549161c	Male	67.0	A	
1730	d8ab900	Female	44.0	B	
1731	e6ff6b3b	Male	67.0	B	
1732	f4f02144	Female	52.0	B	
1733	f5964c51	Female	58.0	A	

1734 rows × 15174 columns

2) Converting Categorical Data to Numerical Data using LabelEncoder

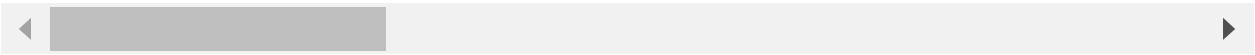
In [17]:
```
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

In [18]:
```
#make a copy
df4_target_biomarker_json_2c=df4_target_biomarker_json
df4_target_biomarker_json_2c
```

Out[18]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.comorbic
0	11156e14a	Male	61.0	A	
1	113d8066d	Male	62.0	B	
2	113ec3f1	Male	59.0	A	
3	114a37875	Male	68.0	B	
4	11b1d32a1	Male	47.0	A	
...	...	...	...	...	
1729	d549161c	Male	67.0	A	
1730	d8ab900	Female	44.0	B	
1731	e6ff6b3b	Male	67.0	B	
1732	f4f02144	Female	52.0	B	
1733	f5964c51	Female	58.0	A	

1734 rows × 15174 columns



In [19]:
```
# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
df4_target_biomarker_json_2c['demographics.gender'] = labelencoder.fit_transform(
df4_target_biomarker_json_2c['status.disease_sub_type'] = labelencoder.fit_transf
df4_target_biomarker_json_2c['status.cohort_qualifier'] = labelencoder.fit_transf
df4_target_biomarker_json_2c['status.smoking_status'] = labelencoder.fit_transfor
df4_target_biomarker_json_2c['status.months_since_diagnosis'] = labelencoder.fit_
df4_target_biomarker_json_2c['demographics.race'] = labelencoder.fit_transform(df
df4_target_biomarker_json_2c['status.days_since_diagnosis'] = labelencoder.fit_tr
df4_target_biomarker_json_2c['status.alcohol_usage'] = labelencoder.fit_transform
df4_target_biomarker_json_2c['status.exercise_frequency'] = labelencoder.fit_trar
df4_target_biomarker_json_2c['status.bmi_level'] = labelencoder.fit_transform(df4
df4_target_biomarker_json_2c['institution'] = labelencoder.fit_transform(df4_targ
df4_target_biomarker_json_2c
```

Out[19]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.comorbid
0	11156e14a	3	61.0	0	
1	113d8066d	3	62.0	1	
2	113ec3f1	3	59.0	0	
3	114a37875	3	68.0	1	
4	11b1d32a1	3	47.0	0	
...	...	...	...	...	
1729	d549161c	3	67.0	0	
1730	d8ab900	1	44.0	1	
1731	e6ff6b3b	3	67.0	1	
1732	f4f02144	1	52.0	1	
1733	f5964c51	1	58.0	0	

1734 rows × 15174 columns

In [20]:
```
#Detect existing (non-missing) values

notNN=df4_target_biomarker_json_2c
notNN= notNN[notNN['demographics.gender'].notna()]
notNN= notNN[notNN['status.disease_sub_type'].notna()]
notNN= notNN[notNN['status.cohort_qualifier'].notna()]
notNN= notNN[notNN['status.smoking_status'].notna()]
notNN= notNN[notNN['status.months_since_diagnosis'].notna()]
notNN= notNN[notNN['status.comorbidity_index'].notna()]
notNN= notNN[notNN['demographics.race'].notna()]
notNN= notNN[notNN['status.days_since_diagnosis'].notna()]
notNN= notNN[notNN['status.alcohol_usage'].notna()]
notNN= notNN[notNN['status.exercise_frequency'].notna()]
notNN= notNN[notNN['status.bmi_level'].notna()]

notNN
```

Out[20]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.comorbic
0	11156e14a	3	61.0	0	
5	11c6138f1	1	65.0	0	
8	11d1ba420	3	29.0	0	
12	127b97df1	3	59.0	0	
13	12a0f9b3b	3	68.0	0	
...	...	...	...	...	
1696	18df819e8	1	39.0	0	
1700	1bbd2af88	1	71.0	0	
1706	202bc2119	3	34.0	0	
1708	229576327	3	63.0	1	
1719	5783682b	1	50.0	0	

353 rows × 15174 columns

In [21]:
```
#some columns have nan data more than half number of data

s=df4_target_biomarker_json_2c
s['patient_id'].isna().sum()                #0
s['demographics.gender'].isna().sum()       #0
s['status.disease_sub_type'].isna().sum()  #0
s['status.cohort_qualifier'].isna().sum()  #0
s['status.smoking_status'].isna().sum()  #0
s['status.months_since_diagnosis'].isna().sum()  #0
s['status.months_since_diagnosis'].isna().sum()  #0
s['demographics.race'].isna().sum()  #0
s['status.days_since_diagnosis'].isna().sum()  #0
s['status.alcohol_usage'].isna().sum()  #0
s['status.exercise_frequency'].isna().sum()  #0
s['status.bmi_level'].isna().sum()  #0
s['status.bmi_level'].isna().sum()  #0
s['institution'].isna().sum()  #0
s['demographics.age'].isna().sum()      #123
s['status.comorbidity_index'].isna().sum() #1381
```

Out[21]:
```
1381
```

3) Data cleaning (filling missing values and droping some columns)

In [25]:
```
#Detect missing values.
s.isna().sum()
```

Out[25]:
```
patient_id                     0
demographics.gender            0
demographics.age             123
status.disease_sub_type        0
status.comorbidity_index    1381
                            ...
BM15153                        0
BM15154                      696
BM15155                        0
BM15156                        0
target_label                   0
Length: 15174, dtype: int64
```

In [26]:
```
def missing_zero_values_table(df):
        zero_val = (df == 0.00).astype(int).sum(axis=0)
        mis_val = df.isnull().sum()
        mis_val_percent = 100 * df.isnull().sum() / len(df)
        mz_table = pd.concat([zero_val, mis_val, mis_val_percent], axis=1)
        mz_table = mz_table.rename(
        columns = {0 : 'Zero Values', 1 : 'Missing Values', 2 : '% of Total Value
        mz_table['Total Zero Missing Values'] = mz_table['Zero Values'] + mz_tabl
        mz_table['% Total Zero Missing Values'] = 100 * mz_table['Total Zero Miss
        mz_table['Data Type'] = df.dtypes
        mz_table = mz_table[
            mz_table.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        print ("Your selected dataframe has " + str(df.shape[1]) + " columns and
            "There are " + str(mz_table.shape[0]) +
              " columns that have missing values.")
#         mz_table.to_excel('D:/sampledata/missing_and_zero_values.xlsx', freeze_
        return mz_table

missing_zero_values_table(s)
```

```
Your selected dataframe has 15174 columns and 1734 Rows.
There are 48 columns that have missing values.
```

Out[26]:

	Zero Values	Missing Values	% of Total Values	Total Zero Missing Values	% Total Zero Missing Values	Data Type
status.comorbidity_index	173	1381	79.6	1554	89.6	float64
BM09176	782	696	40.1	1478	85.2	float64
BM09912	252	696	40.1	948	54.7	float64
BM09932	718	696	40.1	1414	81.5	float64
BM10396	51	696	40.1	747	43.1	float64
BM11007	432	696	40.1	1128	65.1	float64
BM11523	14	696	40.1	710	40.9	float64
BM11929	808	696	40.1	1504	86.7	float64
BM12011	931	696	40.1	1627	93.8	float64
BM12018	155	696	40.1	851	49.1	float64
BM12119	690	696	40.1	1386	79.9	float64
BM12159	58	696	40.1	754	43.5	float64
BM12352	356	696	40.1	1052	60.7	float64
BM12379	302	696	40.1	998	57.6	float64
BM12596	681	696	40.1	1377	79.4	float64
BM12793	318	696	40.1	1014	58.5	float64
BM13623	941	696	40.1	1637	94.4	float64
BM13691	820	696	40.1	1516	87.4	float64
BM13725	826	696	40.1	1522	87.8	float64

	Zero Values	Missing Values	% of Total Values	Total Zero Missing Values	% Total Zero Missing Values	Data Type
BM14027	391	696	40.1	1087	62.7	float64
BM14430	876	696	40.1	1572	90.7	float64
BM14622	1001	696	40.1	1697	97.9	float64
BM15091	85	696	40.1	781	45.0	float64
BM09302	99	696	40.1	795	45.8	float64
BM15154	18	696	40.1	714	41.2	float64
BM09067	1	696	40.1	697	40.2	float64
BM05948	1038	696	40.1	1734	100.0	float64
BM01069	109	696	40.1	805	46.4	float64
BM01254	620	696	40.1	1316	75.9	float64
BM01671	219	696	40.1	915	52.8	float64
BM02181	253	696	40.1	949	54.7	float64
BM02498	704	696	40.1	1400	80.7	float64
BM03527	78	696	40.1	774	44.6	float64
BM05569	906	696	40.1	1602	92.4	float64
BM05796	483	696	40.1	1179	68.0	float64
BM05946	758	696	40.1	1454	83.9	float64
BM05998	254	696	40.1	950	54.8	float64
BM08828	1038	696	40.1	1734	100.0	float64
BM06148	694	696	40.1	1390	80.2	float64
BM06675	931	696	40.1	1627	93.8	float64
BM06893	453	696	40.1	1149	66.3	float64
BM07163	588	696	40.1	1284	74.0	float64
BM07181	825	696	40.1	1521	87.7	float64
BM08310	605	696	40.1	1301	75.0	float64
BM08420	273	696	40.1	969	55.9	float64
BM08498	769	696	40.1	1465	84.5	float64
BM08803	714	696	40.1	1410	81.3	float64
demographics.age	0	123	7.1	123	7.1	float64

In [27]:
```
#drop threshold by column
s=df4_target_biomarker_json_2c
Thresh = 1611 # no null value require, you can also get the by int(x% * len(df))
dfdropCol= s.dropna(thresh = Thresh, axis = 1)
dfdropCol
```

Out[27]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.cohort_q
0	11156e14a	3	61.0	0	
1	113d8066d	3	62.0	1	
2	113ec3f1	3	59.0	0	
3	114a37875	3	68.0	1	
4	11b1d32a1	3	47.0	0	
...	...	...	...	...	
1729	d549161c	3	67.0	0	
1730	d8ab900	1	44.0	1	
1731	e6ff6b3b	3	67.0	1	
1732	f4f02144	1	52.0	1	
1733	f5964c51	1	58.0	0	

1734 rows × 15127 columns

fill age by mean or RandomForestRegressor

In [28]:
```
# Populate Age_Fill

#here I used mean for age regarding gender and race
```

In [29]:
```
dfdropColMeans = dfdropCol.groupby(['demographics.race', 'demographics.gender'])

dfdropCol2=dfdropCol

def f(x):
    if not np.isnan(x["demographics.age"]):  # not NaN
        return x["demographics.age"]
    return dfdropColMeans[x["demographics.race"], x["demographics.gender"]]

dfdropCol2['demographics.age'] = dfdropCol.apply(f, axis=1)

dfdropCol2
```

```
<ipython-input-29-7161a45cd3f7>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  dfdropCol2['demographics.age'] = dfdropCol.apply(f, axis=1)
```

Out[29]:

	patient_id	demographics.gender	demographics.age	status.disease_sub_type	status.cohort_q
0	11156e14a	3	61.0	0	
1	113d8066d	3	62.0	1	
2	113ec3f1	3	59.0	0	
3	114a37875	3	68.0	1	
4	11b1d32a1	3	47.0	0	
...	...	...	...	...	
1729	d549161c	3	67.0	0	
1730	d8ab900	1	44.0	1	
1731	e6ff6b3b	3	67.0	1	
1732	f4f02144	1	52.0	1	
1733	f5964c51	1	58.0	0	

1734 rows × 15127 columns

In [30]:
```
missing_zero_values_table(dfdropCol)
```

```
Your selected dataframe has 15127 columns and 1734 Rows.
There are 0 columns that have missing values.
```

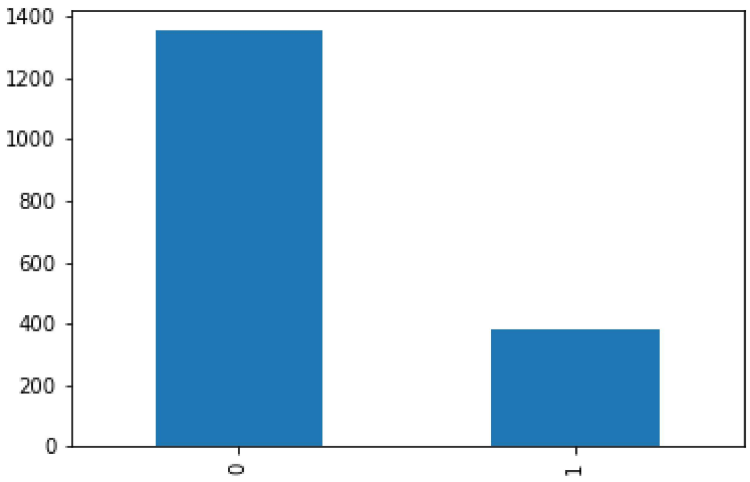
Out[30]:

	Zero Values	Missing Values	% of Total Values	Total Zero Missing Values	% Total Zero Missing Values	Data Type

4) Balanced or imbalanced data

In [32]:
```
# check if data is balance or imbalance
dfdropCol['target_label'].value_counts().plot(kind = 'bar')
```

Out[32]: <AxesSubplot:>



In [33]:
```
# model evaluation
from sklearn.model_selection import train_test_split, cross_val_score, GridSearch
from sklearn.metrics import *

##patient_id, cohort_id, biomarker_id
dfdropCol_features=dfdropCol
dfdropCol_features= dfdropCol_features.drop(['patient_id'],axis=1)
dfdropCol_features= dfdropCol_features.drop(['cohort_id'],axis=1)
dfdropCol_features= dfdropCol_features.drop(['biomarker_id'],axis=1)

dfdropCol_f=dfdropCol_features
dfdropCol_f
```

Out[33]:

	demographics.gender	demographics.age	status.disease_sub_type	status.cohort_qualifier	stat
0	3	61.0	0	0	
1	3	62.0	1	0	
2	3	59.0	0	0	
3	3	68.0	1	0	
4	3	47.0	0	0	
...	...	...	...	...	
1729	3	67.0	0	0	
1730	1	44.0	1	0	
1731	3	67.0	1	0	
1732	1	52.0	1	0	
1733	1	58.0	0	0	

1734 rows × 15124 columns

5) split data to test and train and check balance or imbalance on train and test

In [45]:
```
X = dfdropCol_f.drop(['target_label'],axis=1)
y = dfdropCol_f.target_label
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

In [46]:
```
y.value_counts()
```

Out[46]:
```
0    1353
1     381
Name: target_label, dtype: int64
```

In [47]:
```
y_train.value_counts()
```

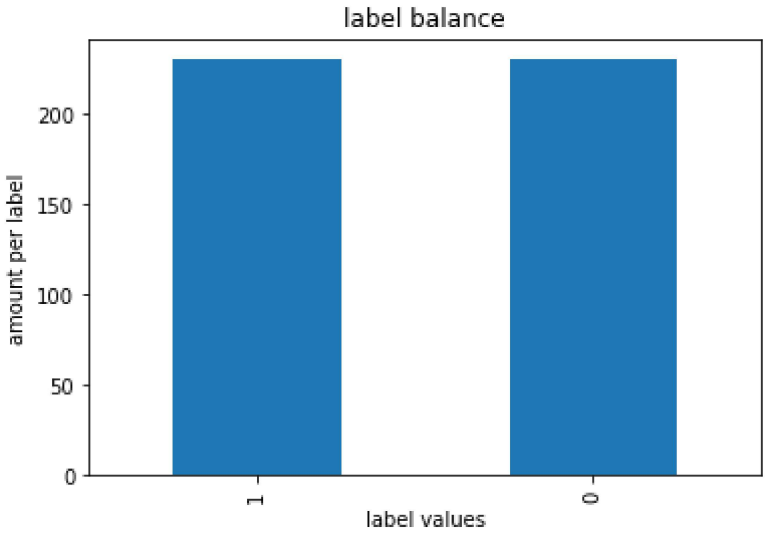
Out[47]:
```
0    1010
1     290
Name: target_label, dtype: int64
```

In [48]:
```
pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\rozita\anaconda3\lib\site-p
ackages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\rozita\anaconda3\li
b\site-packages (from imblearn) (0.9.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\rozita\appdata
\roaming\python\python38\site-packages (from imbalanced-learn->imblearn) (2.2.
0)
Requirement already satisfied: scikit-learn>=1.0.1 in c:\users\rozita\anaconda3
\lib\site-packages (from imbalanced-learn->imblearn) (1.0.2)
Requirement already satisfied: numpy>=1.14.6 in c:\users\rozita\appdata\roaming
\python\python38\site-packages (from imbalanced-learn->imblearn) (1.21.2)
Requirement already satisfied: joblib>=0.11 in c:\users\rozita\appdata\roaming
\python\python38\site-packages (from imbalanced-learn->imblearn) (1.0.1)
Requirement already satisfied: scipy>=1.1.0 in c:\users\rozita\appdata\roaming
\python\python38\site-packages (from imbalanced-learn->imblearn) (1.7.1)
Note: you may need to restart the kernel to use updated packages.
```

In [49]:
```
#balance data , undersampling

from imblearn.datasets import make_imbalance
X_train_1, y_train_1 = make_imbalance(X_train, y_train, sampling_strategy={0:230,
```

In [50]:
```
import matplotlib.pyplot as plt
%matplotlib inline

#balance data
y_train_1.value_counts().plot(kind='bar')
plt.title('label balance')
plt.xlabel('label values')
plt.ylabel('amount per label')
plt.show()
```



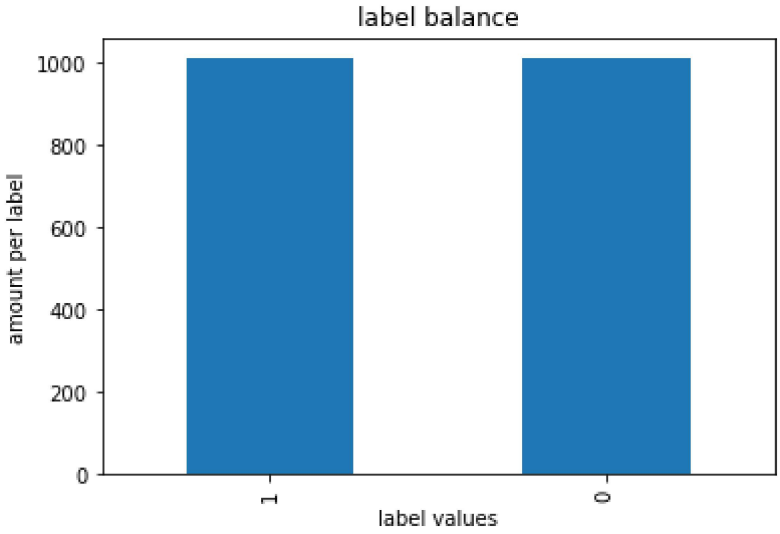
In [51]:
```
# balance data using upsampling
from sklearn.utils import resample

def upsample_classes(data, target):

    lst = list(data[target].unique())

    classes = []
    for c in lst:
        classes.append(data[data[target]==c])

    length = 0
    class_lab = None
    for c in classes:
        if len(c)>length:
            length=len(c)
            class_lab = c
    class_lab = class_lab[target].unique()[0]

    regroup = pd.concat(classes)
    maj_class = regroup[regroup[target]==class_lab]

    lst.remove(class_lab)

    new_classes=[]
    for i in lst:
        new_classes.append(resample(data[data[target]==i],replace=True, n_samples

    minority_classes = pd.concat(new_classes)
    upsample = pd.concat([regroup[regroup[target]==class_lab],minority_classes])

    return upsample
```

In [52]:
```
train = pd.concat([X_train,y_train],axis=1)
train_balanced = (upsample_classes(train,'target_label'))
X_train_2 = train_balanced.drop(['target_label'],axis=1)
y_train_2 = train_balanced.target_label
```

In [53]:
```
y_train_2.value_counts()
```

Out[53]:
```
1    1010
0    1010
Name: target_label, dtype: int64
```

In [54]:
```
y_train_2.value_counts().plot(kind='bar')
plt.title('label balance')
plt.xlabel('label values')
plt.ylabel('amount per label')
plt.show()
```



In [55]:
```
# approach 3 for upsampling is smote
#from imblearn.over_sampling import SMOTE
#smote = SMOTE(random_state = 14)
#X_train_3, y_train_3 = smote.fit_sample(X_train, y_train)
#y_train_3.value_counts().plot(kind='bar')
#plt.title('label balance')
#plt.xlabel('label values')
#plt.ylabel('amount per label')
#plt.show()
```

6) Model

In [94]:
```
pip install xgboost
```

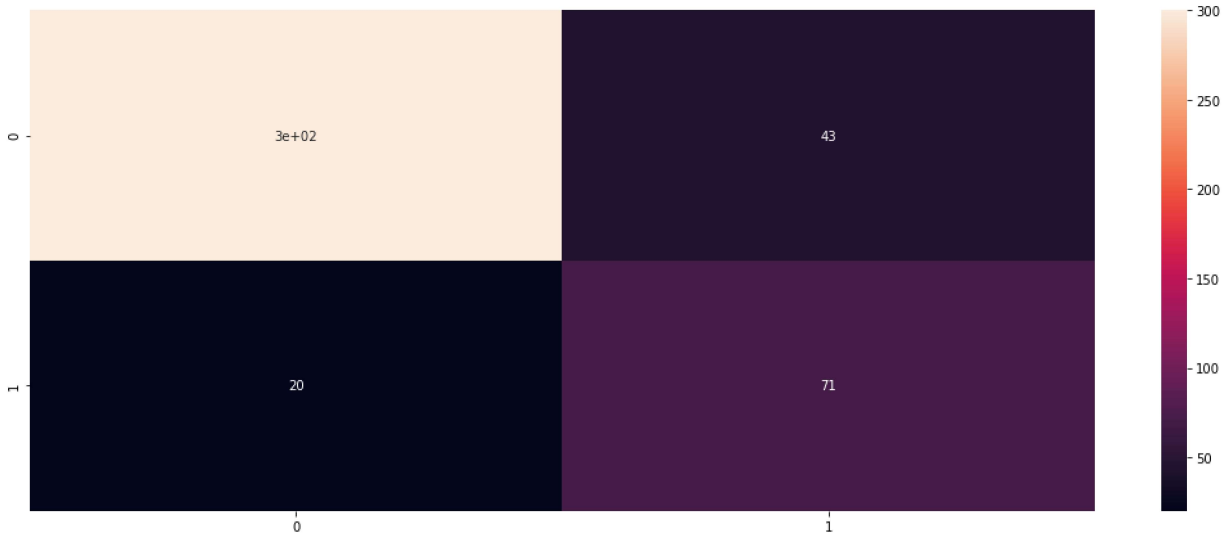
```
Requirement already satisfied: xgboost in c:\users\rozita\anaconda3\lib\site-pa
ckages (1.5.2)
Requirement already satisfied: numpy in c:\users\rozita\appdata\roaming\python
\python38\site-packages (from xgboost) (1.21.2)
Requirement already satisfied: scipy in c:\users\rozita\appdata\roaming\python
\python38\site-packages (from xgboost) (1.7.1)
Note: you may need to restart the kernel to use updated packages.
```

In [61]:
```
# machine learning
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier

import seaborn as sns
from scipy import stats

def classification(X_tr,y_tr,X_te,y_te,method):

    method.fit(X_tr,y_tr)
    p_train = method.predict(X_tr)
    p_test = method.predict(X_te)

    print('train f1 score')
    print(f1_score(y_tr, p_train ,average='micro'))
    print('test f1 score')
    print(f1_score(y_te, p_test ,average='micro'))
    print('-'*20)

    plt.figure(figsize=(15,6))
    sns.heatmap(confusion_matrix(y_te,p_test),annot=True)
    plt.tight_layout()
    plt.show()
```

Evaluation of Model

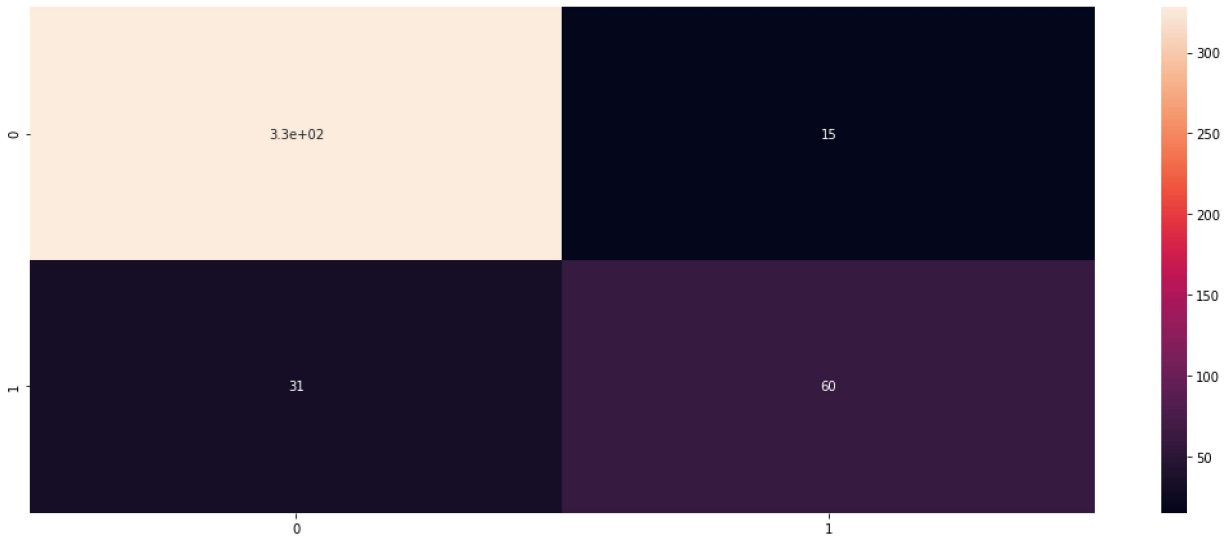
In [63]:
```
xgbc=XGBClassifier(random_state=14,use_label_encoder=False)
method=xgbc
classification(X_train_1,y_train_1,X_test,y_test,method)
```

```
[13:59:21] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.
1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
train f1 score
1.0
test f1 score
0.8548387096774194
--------------------
```



In [65]:
```
xgbc=XGBClassifier(random_state=14,use_label_encoder=False)
method=xgbc
classification(X_train_2,y_train_2,X_test,y_test,method)
```

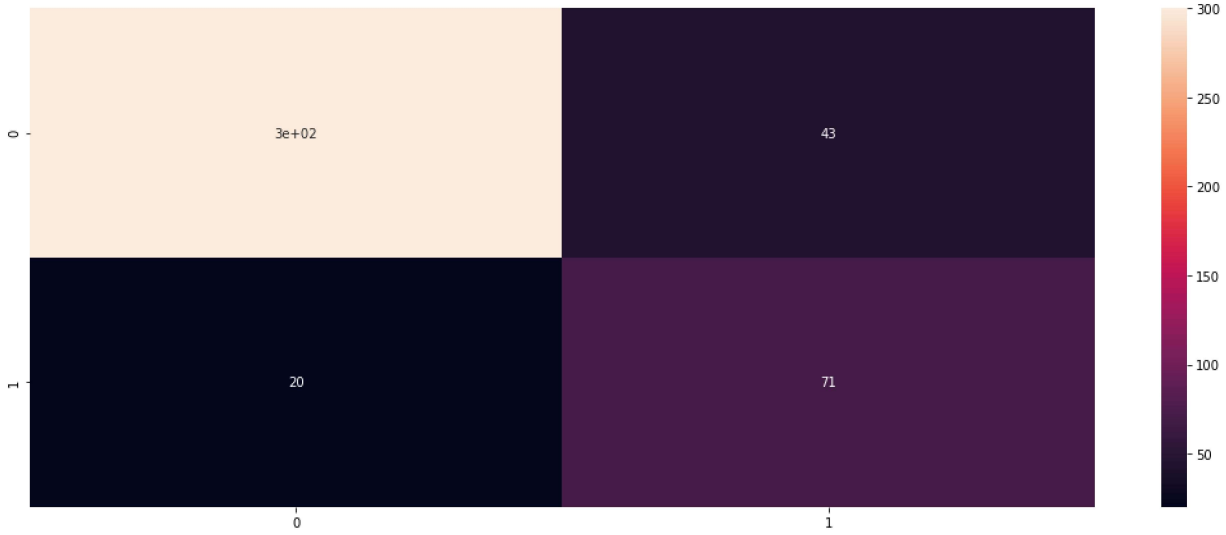
```
[14:20:59] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.
1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
train f1 score
1.0
test f1 score
0.8940092165898618
--------------------
```



In [66]:
```
#balance test data
test = pd.concat([X_test,y_test],axis=1)
test_balanced = (upsample_classes(test,'target_label'))
X_test_b = test_balanced.drop('target_label',axis=1)
y_test_b = test_balanced.target_label
```

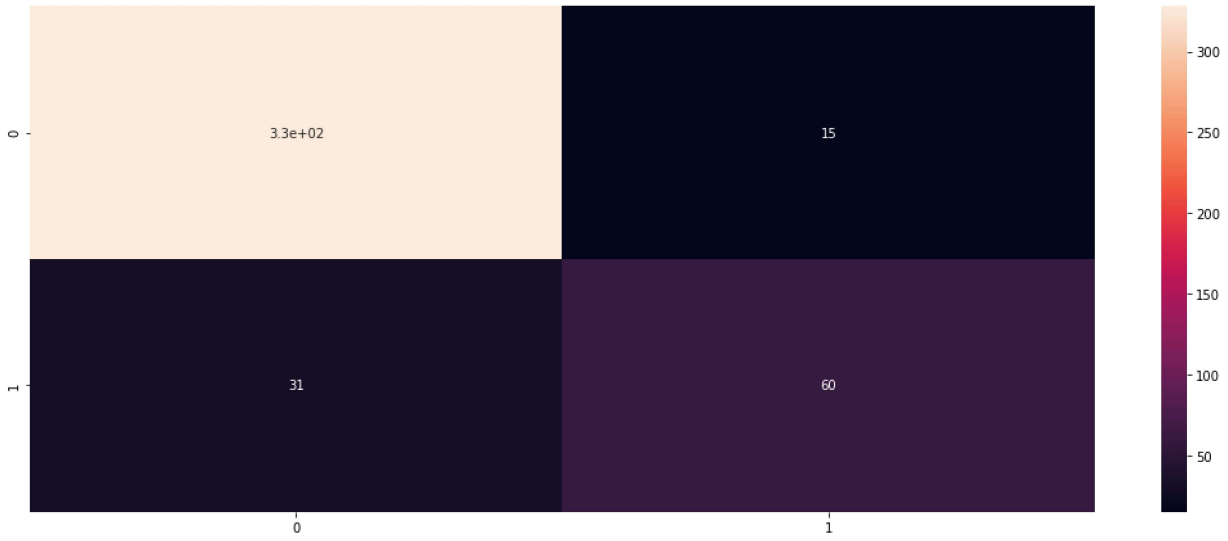
In [67]:
```
classification(X_train_1,y_train_1,X_test,y_test,method)
```

```
[14:21:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.
1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
train f1 score
1.0
test f1 score
0.8548387096774194
--------------------
```



In [68]:
```
classification(X_train_2,y_train_2,X_test,y_test,method)
```

```
[14:22:24] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.
1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
train f1 score
1.0
test f1 score
0.8940092165898618
--------------------
```

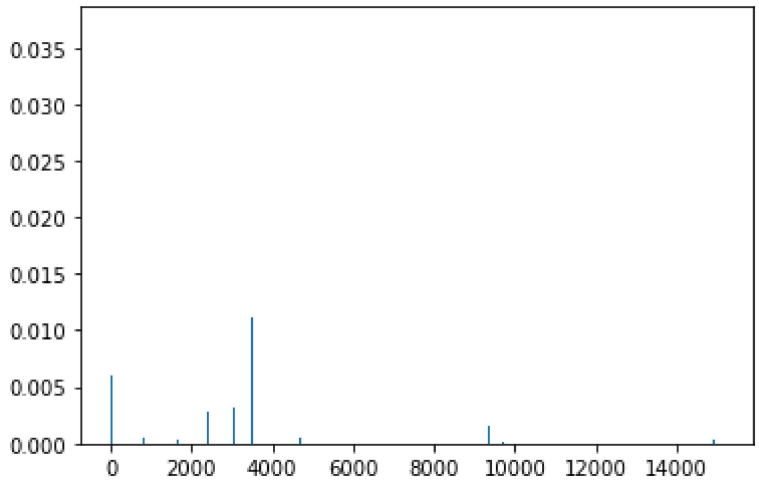


In [69]:
```
len(X_train_1)
```

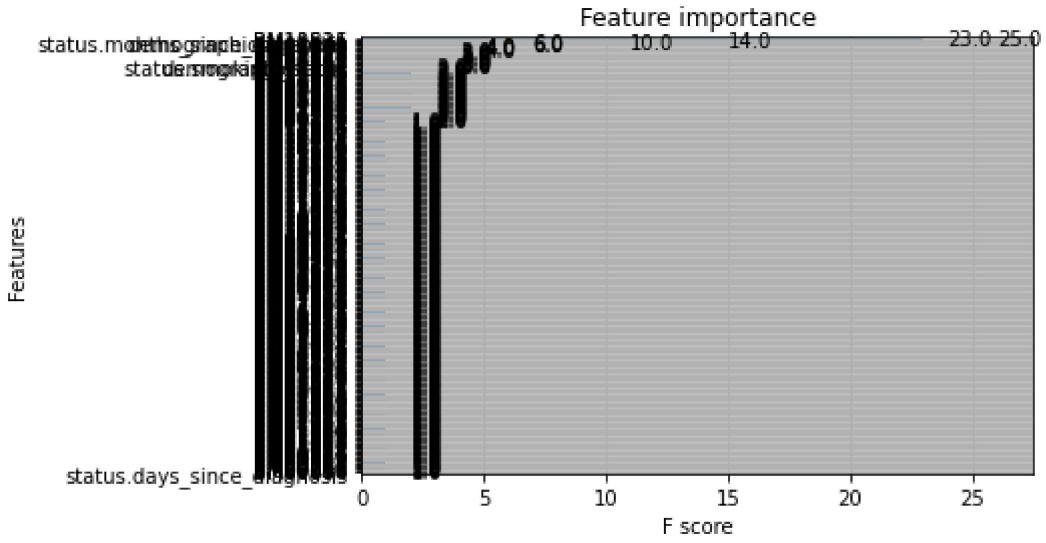
Out[69]:
```
460
```

feature importance

In [73]:
```
# xgboost for feature importance on a classification problem
from sklearn.datasets import make_classification
from xgboost import XGBClassifier
from matplotlib import pyplot
# define dataset
#X, y = make_classification(n_samples=1000, n_features=10, n_informative=5, n_rec
# define the model
model = XGBClassifier(random_state=14,use_label_encoder=False)
# fit the model

XX=X_train_1
yy=y_train_1

model.fit(XX, yy)
# get importance
importance = model.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.10f' % (i,v))
# plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```



In [74]:
```
from xgboost import plot_importance
# plot feature importance
plot_importance(model)
pyplot.show()
```



In [75]:
```
from numpy import sort
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectFromModel

# split data into train and test sets
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random
# fit model on all training data
model = XGBClassifier(random_state=14,use_label_encoder=False)
model.fit(X_train_1, y_train_1)
# make predictions for test data and evaluate
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
# Fit model using each importance as a threshold
thresholds = sort(model.feature_importances_)
thresholds
```

```
[14:34:57] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.
1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy: 85.48%
```

Out[75]:
```
array([0.        , 0.        , 0.        , ..., 0.02350654, 0.02476581,
       0.0367823 ], dtype=float32)
```

In [76]:
```
len(thresholds)
```

Out[76]:
```
15123
```

In [77]:
```
#We can see that the performance of the model generally decreases with the number
for thresh in thresholds:
    # select features using threshold
    if thresh>0.000000000001:
        selection = SelectFromModel(model, threshold=thresh, prefit=True)
        select_X_train = selection.transform(X_train_1)
        # train model
        selection_model = XGBClassifier(random_state=14,use_label_encoder=False)
        selection_model.fit(select_X_train, y_train_1)
        # eval model
        select_X_test = selection.transform(X_test)
        y_pred = selection_model.predict(select_X_test)
        predictions = [round(value) for value in y_pred]
        accuracy = accuracy_score(y_test, predictions)
        print("Thresh=%.3f, n=%d, Accuracy: %.2f%%" % (thresh, select_X_train.sha
```

```
g: X has feature names, but SelectFromModel was fitted without feature names
  warnings.warn(
C:\Users\Rozita\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarnin
g: X has feature names, but SelectFromModel was fitted without feature names
  warnings.warn(

Thresh=0.024, n=3, Accuracy: 88.02%
[14:40:16] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.

C:\Users\Rozita\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarnin
g: X has feature names, but SelectFromModel was fitted without feature names
  warnings.warn(
```

one-hot coding instead of labelencoding

In [ ]: