Importing

In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"r1_wb_lsms_hfpm_hh_survey_public_microdata.csv")
df
```

Out[2]:

	household_id	ii4_resp_id	phw1	cs4_sector	cs5_eaid	cs6_hhid	cs7_hhh_id	cs1_region	cs2_zoneid	cs3_woredaid	...
0	1.01011E+16	1	5860.086426	Rural	10	17	1	Tigray	1	1	...
1	1.01011E+16	1	7373.854980	Rural	10	29	1	Tigray	1	1	...
2	1.01011E+16	1	8227.900391	Rural	10	54	1	Tigray	1	1	...
3	1.01011E+16	1	4227.676758	Rural	10	82	1	Tigray	1	1	...
4	1.01011E+16	1	4918.948242	Rural	10	93	1	Tigray	1	1	...
...	...	...	...	...	...	...	...	...	...	...	...
3244	1.50101E+17	1	2448.561768	Rural	7	101	1	Dire Dawa	1	1	...
3245	1.50101E+17	1	5977.573242	Rural	8	27	1	Dire Dawa	1	1	...
3246	1.50101E+17	2	422.548492	Rural	8	32	1	Dire Dawa	1	1	...
3247	1.50101E+17	1	356.279236	Rural	8	37	1	Dire Dawa	1	1	...
3248	1.50101E+17	3	422.548492	Rural	8	51	1	Dire Dawa	1	1	...

3249 rows × 111 columns

Cleaning and Pre-processing

In [8]:
```
df.head(10)
```

Out[8]:

	household_id	ii4_resp_id	phw1	cs4_sector	cs5_eaid	cs6_hhid	cs7_hhh_id	cs1_region	cs2_zoneid	cs3_woredaid	...	as
0	1.01011E+16	1	5860.086426	Rural	10	17	1	Tigray	1	1	...	
1	1.01011E+16	1	7373.854980	Rural	10	29	1	Tigray	1	1	...	
2	1.01011E+16	1	8227.900391	Rural	10	54	1	Tigray	1	1	...	
3	1.01011E+16	1	4227.676758	Rural	10	82	1	Tigray	1	1	...	
4	1.01011E+16	1	4918.948242	Rural	10	93	1	Tigray	1	1	...	
5	1.01021E+16	1	12665.018550	Rural	10	34	1	Tigray	1	2	...	
6	1.01021E+16	1	13997.995120	Rural	10	35	1	Tigray	1	2	...	
7	1.01021E+16	1	6064.465820	Rural	10	48	1	Tigray	1	2	...	
8	1.01021E+16	1	6064.465820	Rural	10	90	1	Tigray	1	2	...	
9	1.01021E+16	1	14735.886720	Rural	10	104	1	Tigray	1	2	...	

10 rows × 111 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3249 entries, 0 to 3248
Columns: 111 entries, household_id to key
dtypes: float64(3), int64(10), object(98)
memory usage: 2.8+ MB
```

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

Out[4]:

	household_id	ii4_resp_id	phw1	cs4_sector	cs5_eaid	cs6_hhid	cs7_hhh_id	cs1_region	cs2_zoneid	cs3_woredaid	...	as1_as
0	False	False	False	False	False	False	False	False	False	False	...	
1	False	False	False	False	False	False	False	False	False	False	...	
2	False	False	False	False	False	False	False	False	False	False	...	
3	False	False	False	False	False	False	False	False	False	False	...	
4	False	False	False	False	False	False	False	False	False	False	...	
...	...	...	...	...	...	...	...	...	...	...	...	
3244	False	False	False	False	False	False	False	False	False	False	...	
3245	False	False	False	False	False	False	False	False	False	False	...	
3246	False	False	False	False	False	False	False	False	False	False	...	
3247	False	False	False	False	False	False	False	False	False	False	...	
3248	False	False	False	False	False	False	False	False	False	False	...	

3249 rows × 111 columns

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
household_id     0
ii4_resp_id      0
phw1             0
cs4_sector       0
cs5_eaid         0
                ..
ir1_endearly     0
ir_lang          0
ir_understand    0
ir_confident     0
key              0
Length: 111, dtype: int64
```

In [45]:
```
df1=df.dropna()
```

In [32]:
```
df1.describe()
```

Out[32]:

	ii4_resp_id	phw1	cs4_sector	cs5_eaid	cs6_hhid	cs7_hhh_id	cs2_zoneid	cs3_woredaid	cs3c_cityid	cs3c_subcity
count	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000	194.0000
mean	1.376289	9296.943569	1.185567	6.077320	69.216495	1.061856	5.726804	5.329897	6.752577	71.9329
std	1.080997	10343.519691	0.389763	7.879218	55.223376	0.402439	6.572245	4.991644	2.633113	33.7533
min	1.000000	113.589325	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000
25%	1.000000	1344.838226	1.000000	2.000000	27.000000	1.000000	1.000000	2.000000	8.000000	88.0000
50%	1.000000	5088.411133	1.000000	4.000000	55.000000	1.000000	2.000000	5.000000	8.000000	88.0000
75%	1.000000	16668.588867	1.000000	7.750000	95.000000	1.000000	8.000000	7.000000	8.000000	88.0000
max	11.000000	55775.605470	2.000000	71.000000	307.000000	6.000000	35.000000	35.000000	8.000000	88.0000

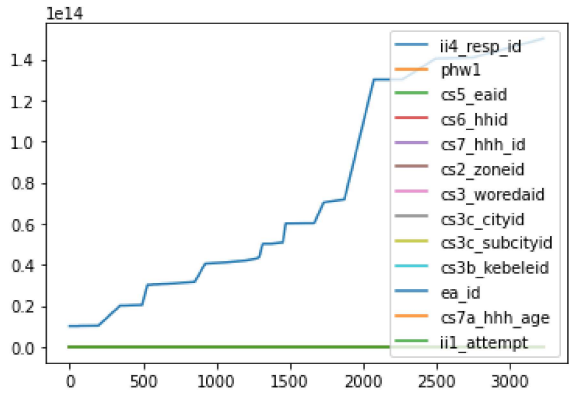
In [12]:
```
df1.columns
```

Out[12]:
```
Index(['household_id', 'ii4_resp_id', 'phw1', 'cs4_sector', 'cs5_eaid',
       'cs6_hhid', 'cs7_hhh_id', 'cs1_region', 'cs2_zoneid', 'cs3_woredaid',
       ...
       'as1_assist_type_3', 'as1_assist_type_0', 'as1_assist_type__98',
       'as1_assist_type__99', 'as1_assist_type__96', 'ir1_endearly', 'ir_lang',
       'ir_understand', 'ir_confident', 'key'],
      dtype='object', length=111)
```

EDA

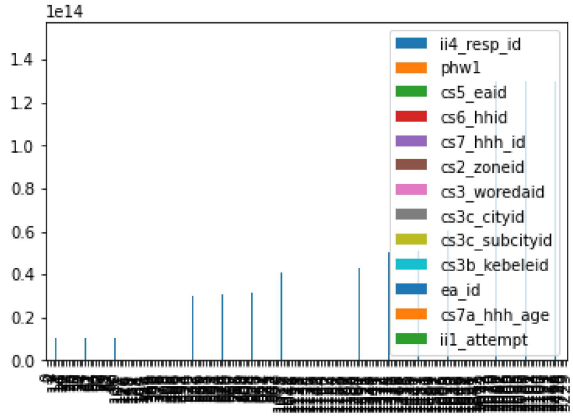
In [14]:
```
df1.plot.line()
```

Out[14]: <AxesSubplot:>



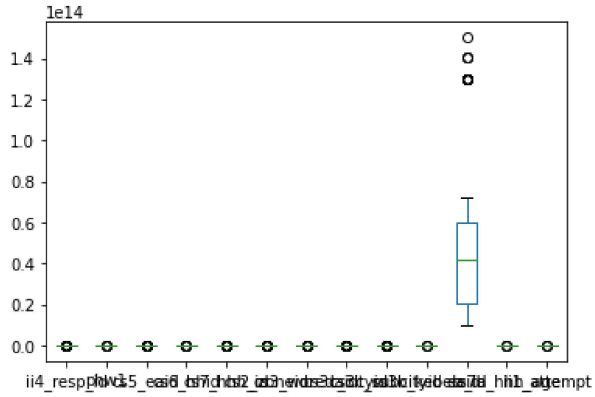
In [15]:
```
df1.plot.bar()
```

Out[15]: <AxesSubplot:>



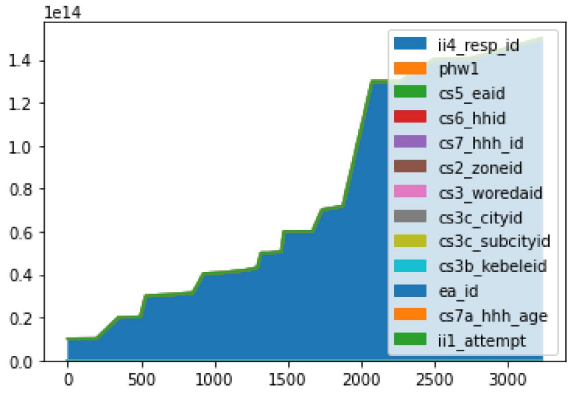
In [16]:
```
df1.plot.box()
```

Out[16]: <AxesSubplot:>



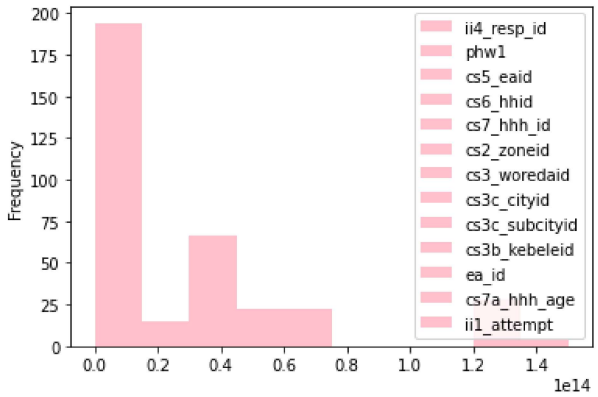
In [17]:
```
df1.plot.area()
```

Out[17]: <AxesSubplot:>



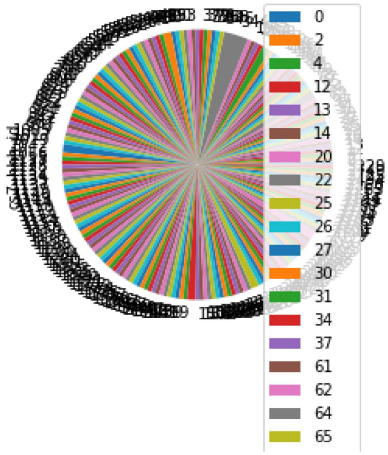
In [18]:
```
df1.plot.hist(color='pink')
```

Out[18]: <AxesSubplot:ylabel='Frequency'>



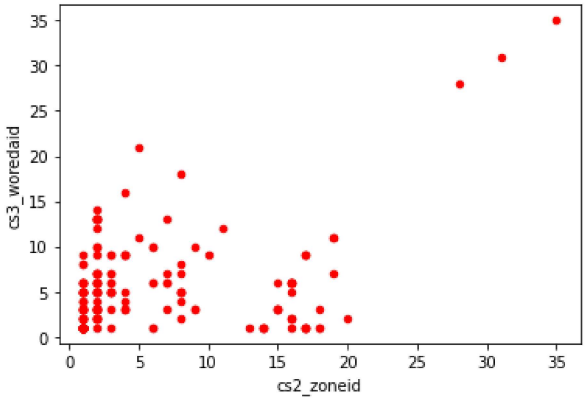
In [22]:
```
df1.plot.pie(y='cs7_hhh_id')
```

Out[22]: <AxesSubplot:ylabel='cs7_hhh_id'>



In [23]:
```
df1.plot.scatter(x='cs2_zoneid',y='cs3_woredaid',color='r')
```

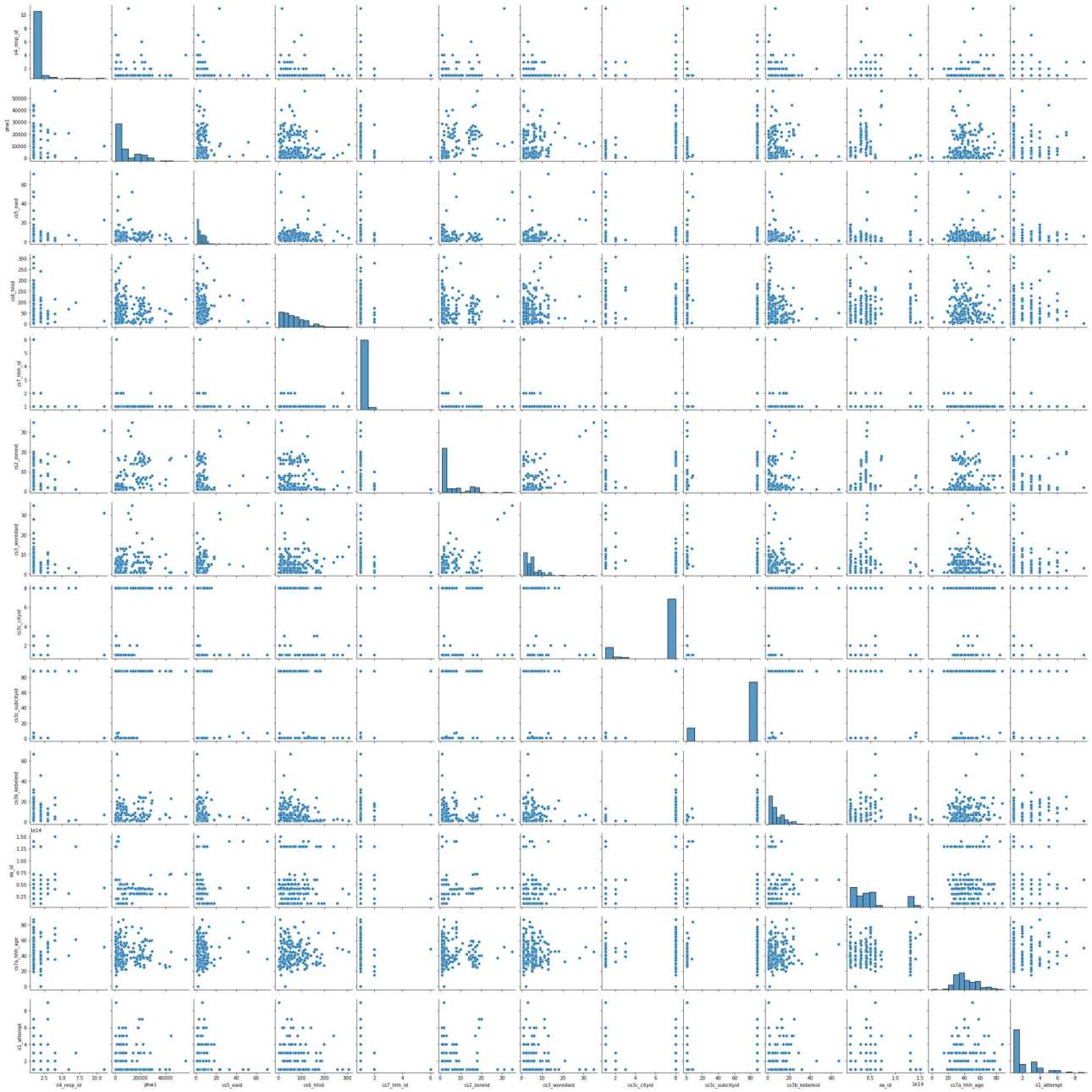
Out[23]: <AxesSubplot:xlabel='cs2_zoneid', ylabel='cs3_woredaid'>



In [24]:
```
df1.to_csv('r1_cleaned.csv')
```

In [25]:
```
sns.pairplot(df1)
```

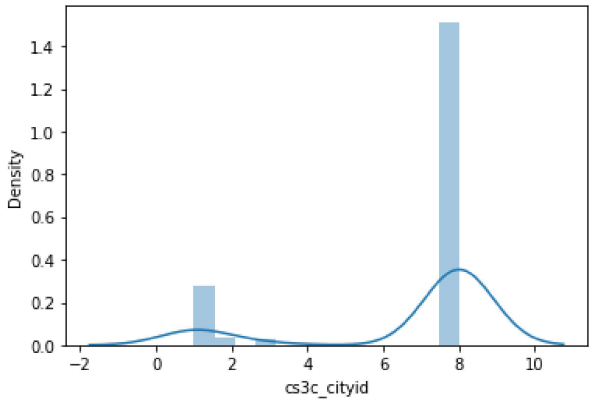
Out[25]: <seaborn.axisgrid.PairGrid at 0x111d21724c0>



In [27]:
```
sns.distplot(df1["cs3c_cityid"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[27]: <AxesSubplot:xlabel='cs3c_cityid', ylabel='Density'>



In [29]:
```
df1["cs4_sector"].value_counts()
```

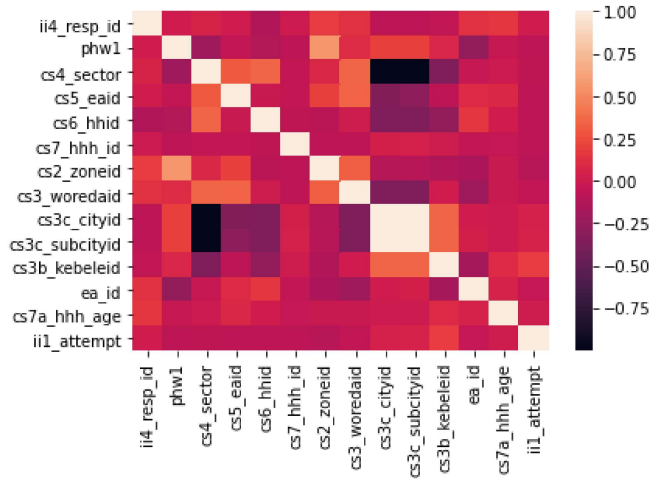
Out[29]:
```
Rural    158
Urban     36
Name: cs4_sector, dtype: int64
```

In [44]:
```
df1=df1.replace({"cs4_sector":{'Rural':1,'Urban':2}})
```

In [33]:
```
dft=df1[['ii4_resp_id','phw1','cs4_sector','cs5_eaid','cs6_hhid','cs7_hhh_id','cs2_zoneid',
         'cs3_woredaid','cs3c_cityid','cs3c_subcityid','cs3b_kebeleid','ea_id','cs7a_hhh_age','ii1_attempt']
```

In [34]:
```
sns.heatmap(dft.corr())
```

Out[34]: <AxesSubplot:>



LinearRegression

In [38]:
```
x=df1[['ii4_resp_id','phw1','cs5_eaid','cs6_hhid','cs7_hhh_id','cs2_zoneid',
       'cs3_woredaid','cs3c_cityid','cs3c_subcityid','cs3b_kebeleid','ea_id','cs7a_hhh_age','ii1_attempt']]
y=df1['cs4_sector']
```

In [39]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[39]:
```
LinearRegression()
```

In [40]:
```
print(lr.intercept_)
```

```
2.010417437787857
```

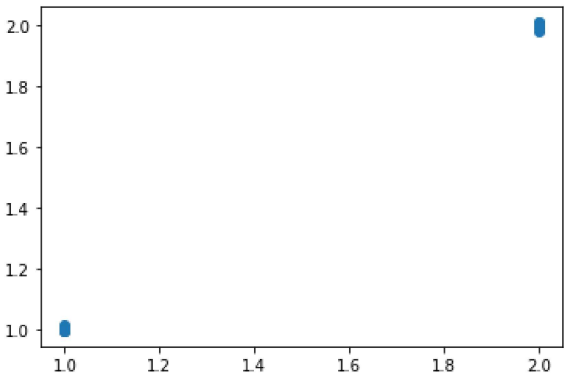
In [41]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[41]:

	Co-efficient
ii4_resp_id	-3.858071e-04
phw1	7.750457e-08
cs5_eaid	6.407573e-04
cs6_hhid	-2.491927e-06
cs7_hhh_id	2.565738e-03
cs2_zoneid	-1.332547e-04
cs3_woredaid	-3.645345e-04
cs3c_cityid	-2.721245e-03
cs3c_subcityid	-1.132377e-02
cs3b_kebeleid	9.342732e-05
ea_id	3.071102e-17
cs7a_hhh_age	5.893967e-05
ii1_attempt	-1.937856e-04

In [42]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[42]:
```
<matplotlib.collections.PathCollection at 0x111dcc1fd90>
```



In [43]:
```
print(lr.score(x_test,y_test))
```

```
0.999828527346587
```

Ridge & Lasso

In [46]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [47]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:147: LinAlgWarning: Ill-condition
ed matrix (rcond=5.99407e-29): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True,
```

Out[47]:
```
Ridge(alpha=10)
```

In [48]:
```
rr.score(x_test,y_test)
```

Out[48]:
```
0.9998538266928155
```

In [49]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[49]:
```
Lasso(alpha=10)
```

In [50]:
```
la.score(x_test,y_test)
```

Out[50]:
```
0.41648534607732024
```

LogisticRegression

In [55]:
```
from sklearn.linear_model import LogisticRegression
lgr=LogisticRegression()
lgr.fit(x_train,y_train)
```

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

In [56]:
```
lgr.predict(x_test)
```

Out[56]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [57]:
```
lgr.score(x_test,y_test)
```

Out[57]:
```
0.8305084745762712
```

In [60]:
```
from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(x)
logr=LogisticRegression()
logr.fit(fs,y)
```

Out[60]:
```
LogisticRegression()
```

In [90]:
```
O=[[1,2,3,4,5,6,7,8,9,10,11,12,13]]
Pred=logr.predict(O)
print(Pred)
```

```
[1]
```

In [64]:
```
logr.classes_
```

Out[64]:
```
array([1, 2], dtype=int64)
```

In [91]:
```
logr.predict_proba(O)[0][0]
```

Out[91]:
```
0.9999999999999996
```

In [92]:
```
logr.predict_proba(O)[0][1]
```

Out[92]:
```
4.0165874998915433e-16
```

RandomForest

In [68]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[68]:
```
RandomForestClassifier()
```

In [69]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [70]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[70]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

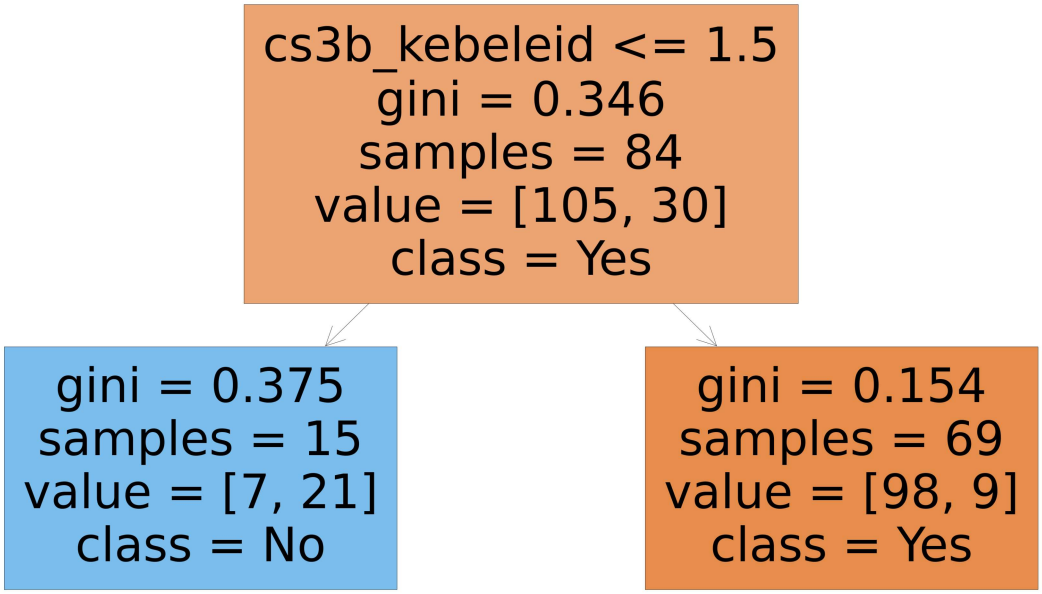
In [71]:
```
grid_search.best_score_
```

Out[71]:
```
1.0
```

In [72]:
```
rfc_best=grid_search.best_estimator_
```

In [73]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True)
```

Out[73]:
```
[Text(2232.0, 1630.8000000000002, 'cs3b_kebeleid <= 1.5\ngini = 0.346\nsamples = 84\nvalue = [105, 30]\ncla
ss = Yes'),
 Text(1116.0, 543.5999999999999, 'gini = 0.375\nsamples = 15\nvalue = [7, 21]\nclass = No'),
 Text(3348.0, 543.5999999999999, 'gini = 0.154\nsamples = 69\nvalue = [98, 9]\nclass = Yes')]
```



ElasticNet

In [75]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[75]:
```
ElasticNet()
```

In [76]:
```
print(en.coef_)
```

```
[ 0.00000000e+00 -3.19818139e-07  0.00000000e+00  0.00000000e+00
 -0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -1.10834148e-02 -0.00000000e+00  1.25736768e-17  0.00000000e+00
 -0.00000000e+00]
```

In [77]:
```
print(en.intercept_)
```

```
1.9852529659555396
```

In [79]:
```
print(en.predict(x_train))
```

```
[1.00765547 1.01030382 1.96978203 1.00434611 1.01030128 0.99744399
 1.97418689 1.00569916 1.0112971  1.97475931 1.00417417 1.97474084
 1.01128145 1.00810184 1.9532518  1.97205789 1.00173033 1.01142809
 1.97281999 1.01135805 1.00412711 1.01138479 1.00417417 1.97469258
 1.00412006 1.97341883 1.0110322  1.00717239 1.00957137 0.99297722
 1.01023251 1.00465665 1.00416339 1.0112971  1.00449451 1.0063399
 1.9733834  1.0084663  1.97141705 1.00812466 1.00305125 1.00786595
 1.00806192 0.99765609 1.01053286 1.00861529 1.00973586 1.00828973
 1.00890011 1.00995112 1.97476852 1.00962978 1.00736308 1.01126638
 1.00945072 1.01135805 0.99900627 1.97244238 1.00311084 1.00220856
 1.00946393 1.00924007 1.97079257 1.97208756 1.01032602 1.9742244
 1.00740804 1.00390702 1.97087186 1.00964517 1.00453636 1.97374149
 1.00964744 1.00368113 1.00539518 1.9733834  1.00313266 1.01138598
 1.00417417 1.0114502  1.0114502  1.00994349 1.00336873 1.00877219
 1.00317447 1.00915251 1.00723546 1.00872202 1.97422336 1.00861529
 1.0071991  1.00857065 1.00847697 1.00962978 1.009156   1.00942353
 1.00861529 1.00917822 1.00860252 1.97211252 1.00803932 1.97573373
 1.00344118 1.0086026  1.00857065 1.01024037 1.0100776  1.01135579
 1.00783605 1.97349545 1.01138479 1.01139141 1.9705029  1.00840613
 1.00467162 1.00305125 1.00941709 1.00730014 1.97575628 1.01151199
 0.99670976 1.00477397 1.00225487 1.00577315 1.00929216 1.01149519
 1.89754954 1.00173033 1.00334072 1.00892502 1.0018932  1.00358103
 1.01142618 1.0114502  1.00967085]
```

In [89]:
```
print(en.score(x_train,y_train))
```

```
0.9982212920770555
```

Best model:RandomForest

In [ ]:
```

```