In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
```

Let's load the Imputed Dataframe to perform further EDA and Feature Engineering.

In [2]:

```
sampleFile = open("imputed_df","rb")
imputed_df = pickle.load(sampleFile)
sampleFile.close()
```

In [3]:

```
## Divide the data into train and test set for further investigation and feature exploration
X = imputed_df.drop(['classification'],axis=1)
y = imputed_df['classification']

from sklearn.model_selection import train_test_split
## Dividing the whole dataset into train and test
X_train_cv,X_test,y_train_cv, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
## Dividing the training set again into train and cross-validation
X_train,X_cv,y_train,y_cv = train_test_split(X_train_cv,y_train_cv, test_size=0.20, random_state=4
2)
```

In [4]:

```
## Dumping important pieces of data so that it can be used for later purposes.
sampleFile = open("X_train","wb")
pickle.dump(X_train,sampleFile)
sampleFile.close()

sampleFile = open("y_train","wb")
pickle.dump(y_train,sampleFile)
sampleFile.close()

sampleFile = open("X_cv","wb")
pickle.dump(X_cv,sampleFile)
sampleFile.close()

sampleFile = open("y_cv","wb")
pickle.dump(y_cv,sampleFile)
sampleFile.close()

sampleFile = open("X_test","wb")
pickle.dump(X_test,sampleFile)
sampleFile.close()
```

Combined the X_train and y_train to form X_train_df
Which in turn will help us to perform Exploratory Data Analysis and Feature Engineering

In [5]:

```
X_train_df = pd.concat([X_train,y_train],axis=1)
```

In [6]:

```
X_train_df.head(5)
```

Out[6]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification

179	179	70.0	70.0	90.0	1.010	2.0	0.0	0	1.0	notpresent	notpresent	notpresent	...	39	8900	3.8	no	no	no	good	no	no	notckd
270	270	23.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	41	7200	5.0	no	no	no	good	no	no	notckd
323	323	43.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	45	7800	4.5	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	53	6800	6.1	no	no	no	good	no	no	notckd
40	40	46.0	90.0	1.010	2.0	0.0	0	1.0	notpresent	notpresent	...	32	9100	4.1	yes	no	\tno	good	no	no	ckd

5 rows × 26 columns

In [7]:

```
len(X_train_df.columns)
```

Out[7]:

```
26
```

In [8]:

```
## Number of features = 25
##Let's do some Univariate Analysis first
```

In [9]:

```
## High Level Analysis
## Shape of dataframe
X_train_df.shape
```

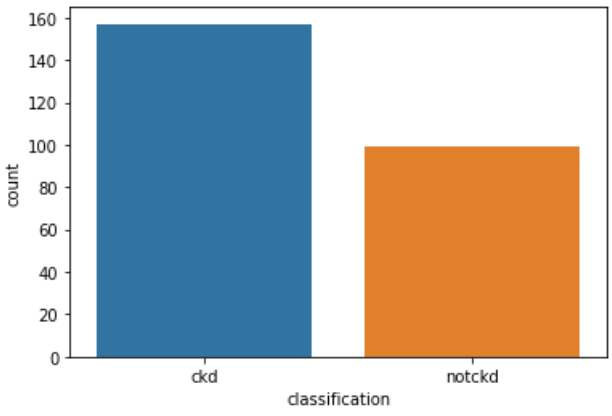
Out[9]:

```
(256, 26)
```

1. Univariate Feature Analysis

In [10]:

```
## Distribution of target variable
ax = sns.countplot(x="classification", data=X_train_df)
```



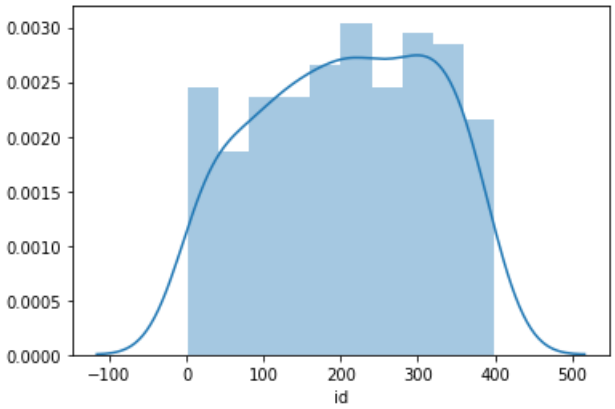
Distribution of target variable seems to fairly balanced.
Now let's do some Univariate feature analysis.

In [11]:

```
sns.distplot(a = X_train_df['id'],bins = 10)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407b71d088>
```



All values are uniformly distributed between 0-400 showing no variance and suggesting that it is a unique identifier.
Though it was evident from the name of the column itself that it will be an unique identifier but still confirming by plotting or seeing value counts is better.

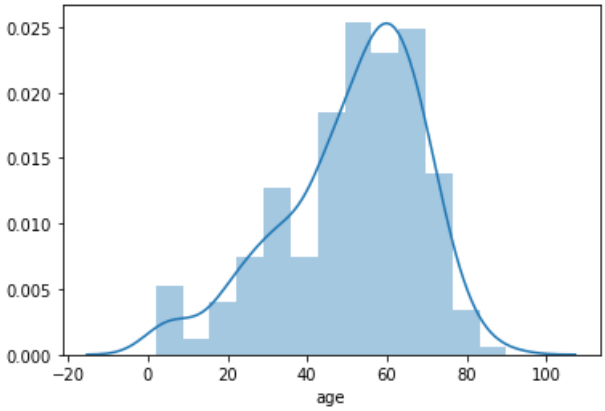
1.1 Analysing feature age

In [12]:

```
sns.distplot(X_train_df['age'], label = 'class == ckd')
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407af71488>
```

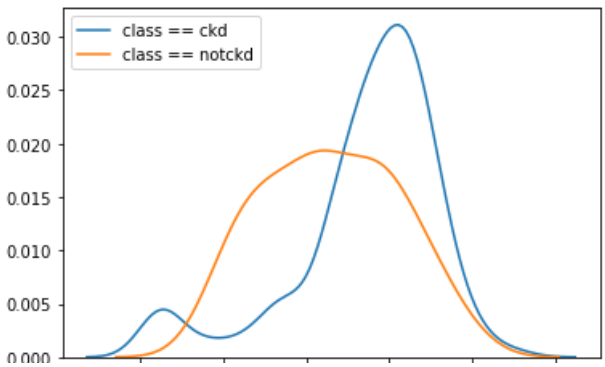


In [13]:

```
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'ckd','age'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'notckd','age'], label = 'class == notck
d')
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407b86a1c8>
```



The above kde plot suggests that people below the age of 40 have less chances of ckd but as a person's age increases there are more chances of a him/her having ckd.
Adding a new column named target which is nothing but a 0/1 representation of target variable 'classification'

In [14]:

```
X_train_df['target'] = X_train_df['classification'].apply(lambda x : 1 if x == 'ckd' else 0)
```

In [15]:

```
X_train_df['target'].head()
X_train_df['target'].value_counts()
```

Out[15]:

```
1    157
0     99
Name: target, dtype: int64
```

In [16]:

```
X_train_df[['age','target']].corr()
```

Out[16]:

	age	target
age	1.000000	0.240848
target	0.240848	1.000000

Correlation with the target variable is very low.
Since .corr() checks for linear relationship.
Let's see what happens if we just bin this feature we have already seen in the kdeplot that the distribution is left skewed.
Let's divide the feature values into 4 bins

- 0-20
- 20-40
- 40-60
- above 60

In [17]:

```
def formBinsForAge(x):
  if x <= 20:
    return 0
  elif x > 20 and x <= 40:
    return 1
  elif x > 40 and x <= 60:
    return 2
  elif x > 60:
    return 3
  else:
    return np.nan
```

In [18]:

```
X_train_df['age_bins'] = X_train_df['age'].apply(lambda x : formBinsForAge(x))
print(X_train_df['age_bins'].value_counts())
X_train_df.head(4)
```

```
2    112
3     85
1     43
0     16
```

```
Name: age_bins, dtype: int64
```

Out[18]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	rc	htn	dm	cad	appet	pe	ane	classification	target	ag
179	179	72.0	90.0	1.010	2.0	0.0	1	1.0	present	notpresent	...	3.9	no	no	no	good	no	no	ckd	1	
270	270	23.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	5.0	no	no	no	good	no	no	notckd	0	
323	323	43.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	4.5	no	no	no	good	no	no	notckd	0	
399	399	58.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	6.1	no	no	no	good	no	no	notckd	0	

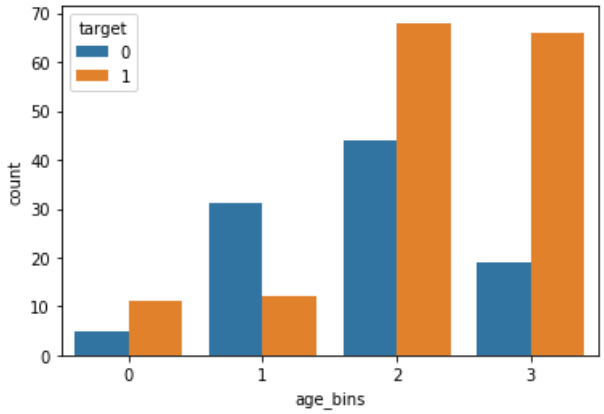
4 rows × 28 columns

In [19]:

```
sns.countplot(x = 'age_bins',hue='target',data=X_train_df)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407b90f948>
```



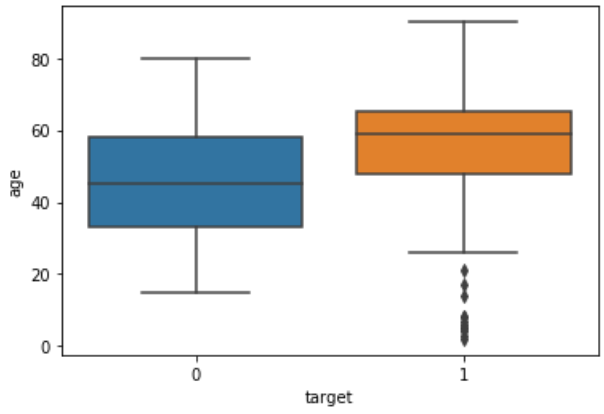
Here the information is more clear that the people in the age group of above 60 are prone to chronic kidney disease.

In [20]:

```
sns.boxplot(x = 'target',y='age',data=X_train_df)
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407b9805c8>
```



Even the boxplots show good seperation capability

1.2 Analysing feature blood pressure

In [21]:

```
X_train_df['bp'].value_counts()
```

Out[21]:

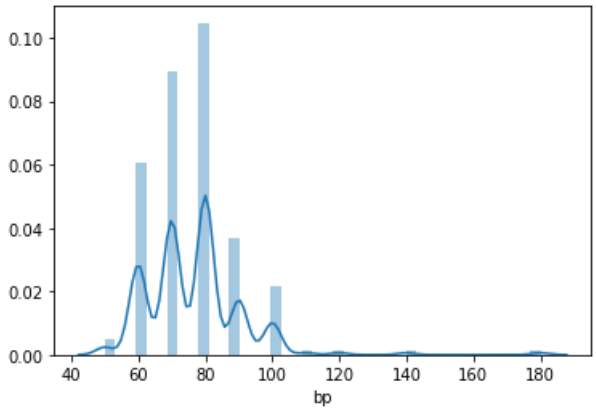
```
80.0     83
70.0     71
60.0     48
90.0     29
100.0    17
50.0      4
120.0     1
180.0     1
140.0     1
110.0     1
Name: bp, dtype: int64
```

In [22]:

```
sns.distplot(X_train_df['bp'])
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407b9bd0c8>
```

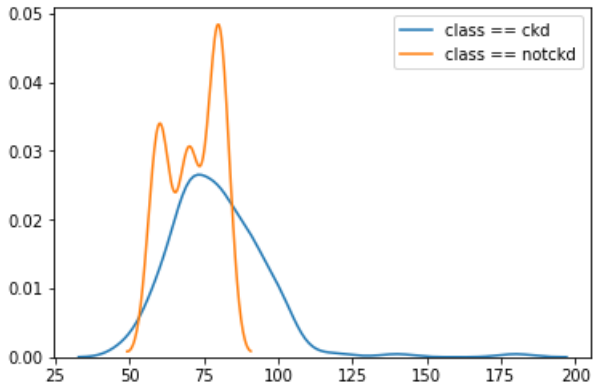


In [23]:

```
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 1,'bp'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 0,'bp'], label = 'class == notckd')
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bad5048>
```



As can be seen from the above plot people with high blood pressure have ckd as compared to people with normal blood pressure range.

In [1]:

```
def formBPBins(x):
  if x <= 60:
     return 0
  elif x > 60 and x <= 80:
    return 1
  elif x > 80  and x <= 90:
    return 2
  elif x > 90:
    return 3
```

In [25]:

```
X_train_df['bp_bins'] = X_train_df['bp'].apply(lambda x : formBPBins(x))
```

In [26]:

```
X_train_df['bp_bins'].value_counts()
```

Out[26]:

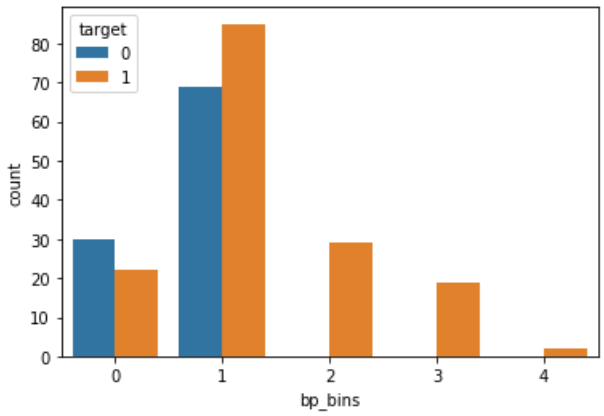
```
1    154
0     52
2     29
3     19
4      2
Name: bp_bins, dtype: int64
```

In [27]:

```
sns.countplot(x = 'bp_bins',hue='target',data=X_train_df)
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bb5f588>
```

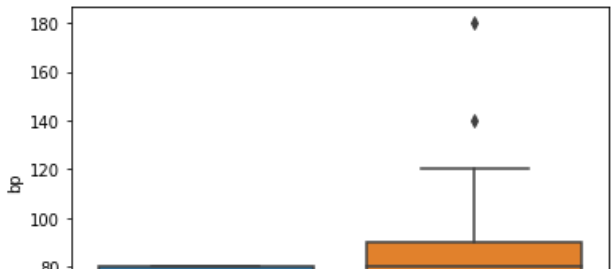


In [28]:

```
sns.boxplot(x = 'target',y='bp',data=X_train_df)
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bbe7648>
```



People in the higher blood pressure range are more prone to CKD as compared to lower blood pressure range.

In [29]:

```
group_df = X_train_df[['age_bins','bp_bins','target']].groupby(['age_bins','bp_bins'])
```

In [30]:

```
X_train_df[['age_bins','bp']].groupby(['age_bins']).get_group('0').mean()
X_train_df[['age_bins','bp']].groupby(['age_bins']).get_group('0').min()
X_train_df[['age_bins','bp']].groupby(['age_bins']).get_group('0').max()
```

Out[30]:

```
age_bins     0.0
bp          80.0
dtype: float64
```

In [31]:

```
aggFunc = ['mean','max','min','count']
for agg in aggFunc:
  nan_count = 0
  for age_bin in X_train_df['age_bins'].unique():
    try:
      if agg=='mean':
        X_train_df.loc[(X_train_df['age_bins']==str(age_bin)),'age_bp_mean'] = X_train_df[['age_bin
s','bp']].groupby(['age_bins']).get_group(str(age_bin)).mean()['bp']
      elif agg=='max':
        X_train_df.loc[(X_train_df['age_bins']==str(age_bin)),'age_bp_max'] = X_train_df[['age_bins
','bp']].groupby(['age_bins']).get_group(str(age_bin)).max()['bp']
      elif agg=='min':
        X_train_df.loc[(X_train_df['age_bins']==str(age_bin)),'age_bp_min'] = X_train_df[['age_bins
','bp']].groupby(['age_bins']).get_group(str(age_bin)).min()['bp']
      elif agg=='count':
        X_train_df.loc[(X_train_df['age_bins']==str(age_bin)),'age_bp_count'] = X_train_df[['age_bi
ns','bp']].groupby(['age_bins']).get_group(str(age_bin)).count()['bp']
    except:
      nan_count += 1
```

In [32]:

```
print(X_train_df['age_bp_mean'].value_counts())
print(X_train_df['age_bp_max'].value_counts())
print(X_train_df['age_bp_min'].value_counts())
print(X_train_df['age_bp_count'].value_counts())
```

```
78.750000    112
76.470588     85
72.790698     43
68.750000     16
Name: age_bp_mean, dtype: int64
100.0    128
180.0    112
80.0      16
Name: age_bp_max, dtype: int64
50.0    128
60.0    128
Name: age_bp_min, dtype: int64
112.0    112
85.0      85
43.0      43
16.0      16
Name: age_bp_count, dtype: int64
```

Since we are going to develop more features instead of adding them to original dataframe, we'll create separate feature set.

In [33]:

```
X_train_df.head()
```

Out[33]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pe	ane	classification	target	age_bins	bp_bins	age_bp_m
179	179	72.0	90.0	1.010	2.0	0.0	1	1.0	present	notpresent	...	no	no	ckd	1	3	2	76.470
270	270	23.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	no	no	notckd	0	1	1	72.790
323	323	43.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	no	no	notckd	0	2	1	78.750
399	399	58.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	no	no	notckd	0	2	1	78.750
40	40	46.0	90.0	1.010	2.0	0.0	0	1.0	notpresent	notpresent	...	no	no	ckd	1	2	2	78.750

5 rows × 33 columns

In [34]:

```
feature_df = pd.DataFrame()
feature_df['target'] = X_train_df['target']
feature_df['age_bins'] = X_train_df['age_bins']
feature_df['bp_bins'] = X_train_df['bp_bins']
feature_df['age_bp_mean'] = X_train_df['age_bp_mean']
feature_df['age_bp_max'] = X_train_df['age_bp_max']
feature_df['age_bp_min'] = X_train_df['age_bp_min']
feature_df['age_bp_count'] = X_train_df['age_bp_count']
```

In [35]:

```
X_train_df.drop(['target','age_bins','bp_bins','age_bp_mean','age_bp_max','age_bp_min','age_bp_cour
t'],axis=1,inplace = True)
X_train_df.head()
```

Out[35]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
179	179	72.0	90.0	1.010	2.0	0.0	1	1.0	present	notpresent	...	39	8900	3.9	no	no	no	good	no	no	ckd
270	270	23.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	41	7200	5.0	no	no	no	good	no	no	notckd
323	323	43.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	45	7800	4.5	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	0	0.0	notpresent	notpresent	...	53	6800	6.1	no	no	no	good	no	no	notckd
40	40	46.0	90.0	1.010	2.0	0.0	0	1.0	notpresent	notpresent	...	32	9100	4.1	yes	no	\tno	good	no	no	ckd

5 rows × 26 columns

In [36]:

```
feature_df.head()
```

Out[36]:

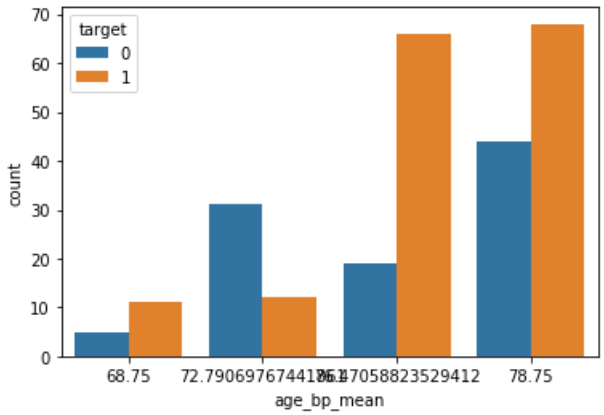
	target	age_bins	bp_bins	age_bp_mean	age_bp_max	age_bp_min	age_bp_count
179	1	3	2	76.470588	100.0	60.0	85.0
270	0	1	1	72.790698	100.0	60.0	43.0
323	0	2	1	78.750000	180.0	50.0	112.0
399	0	2	1	78.750000	180.0	50.0	112.0
40	1	2	2	78.750000	180.0	50.0	112.0

In [37]:

```
sns.countplot(x = 'age_bp_mean',hue='target',data=feature_df)
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bc535c8>
```

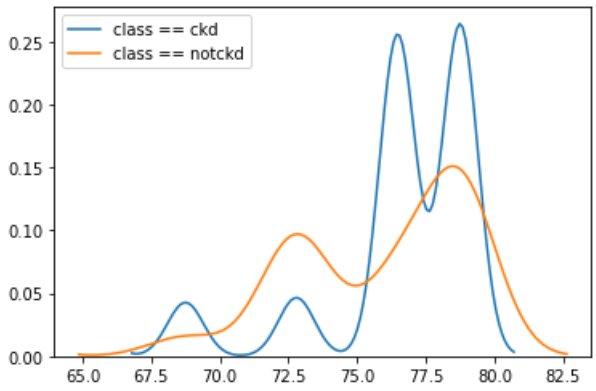


In [38]:

```
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'age_bp_mean'], label = 'class == ckd')
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'age_bp_mean'], label = 'class == notckd')
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bcfbbc8>
```

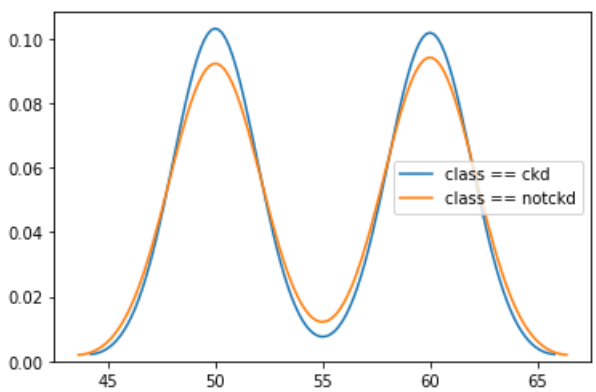


In [39]:

```
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'age_bp_min'], label = 'class == ckd')
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'age_bp_min'], label = 'class == notckd')
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bcf6288>
```



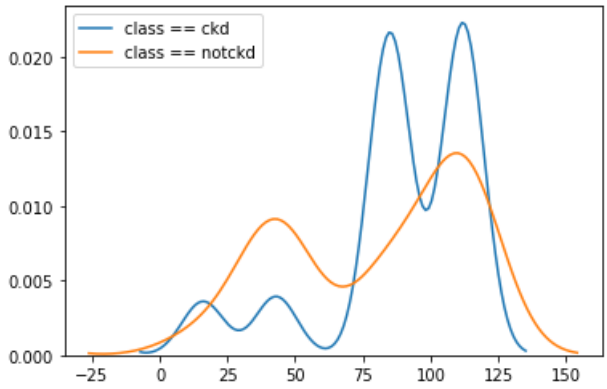
This feature won't be of any use.
Even max count gives similar result.

In [40]:

```
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'age_bp_count'], label = 'class == ckd')
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'age_bp_count'], label = 'class == notckd')
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bb5f608>
```



In [41]:

```
feature_df['age_bins'] = feature_df['age_bins'].apply(lambda x : str(x))
```

In [42]:

```
abs(feature_df.corr()['target'])
```

Out[42]:

```
target          1.000000
bp_bins         0.371661
age_bp_mean     0.117615
age_bp_max      0.015384
age_bp_min      0.008021
age_bp_count    0.133578
Name: target, dtype: float64
```

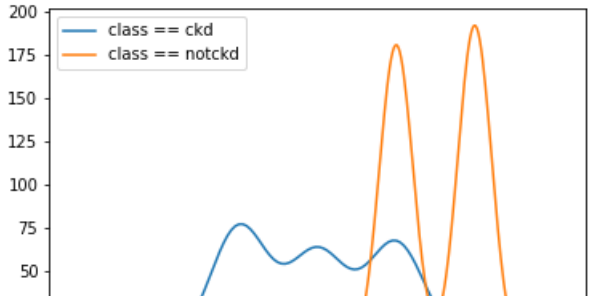
1.3 Analysing feature Specific Gravity(Sg)

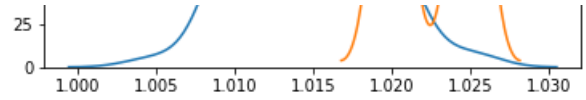
In [43]:

```
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'ckd','sg'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'notckd','sg'], label = 'class == notckd
')
```

Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407bddf888>
```





In [44]:

```
X_train_df['target'] = feature_df['target']
```

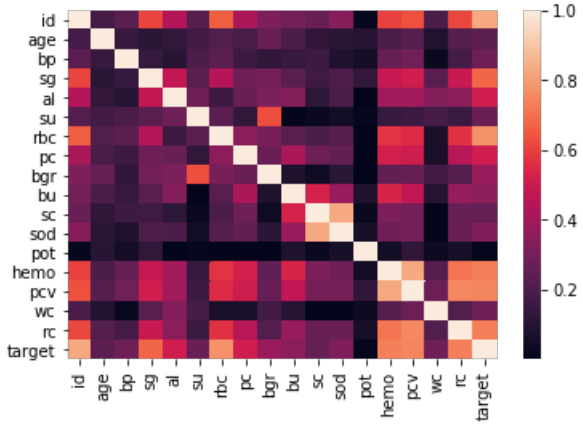
Since going through each and every feature will be cumbersome.
It's better that I filter few features and start building on those features.

In [45]:

```
sns.heatmap(abs(X_train_df.corr()))
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407beeef48>
```



In [46]:

```
abs(X_train_df.corr()['target'])
```

Out[46]:

```
id          0.829134
age         0.240848
bp          0.280729
sg          0.678348
al          0.507968
su          0.273448
rbc         0.775688
pc          0.510749
bgr         0.383836
bu          0.353857
sc          0.258217
sod         0.322329
pot         0.010682
hemo        0.732390
pcv         0.748686
wc          0.279009
rc          0.733150
target      1.000000
Name: target, dtype: float64
```

In [47]:

```
corr_ser = X_train_df.corr()['target']
corr_ser[corr_ser >= 0.4]
```

Out[47]:

```
al          0.507968
```

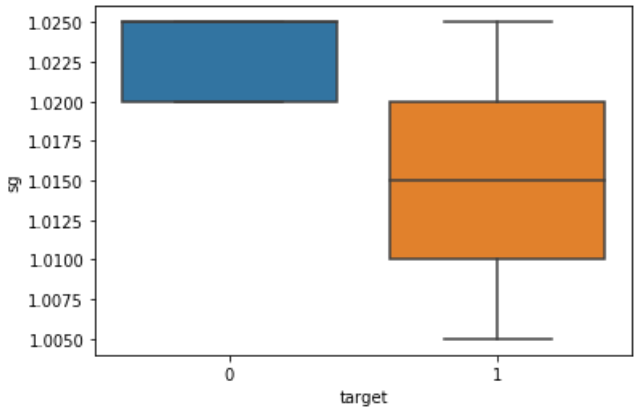
```
rbc       0.775688
pc        0.510749
target    1.000000
Name: target, dtype: float64
```

In [48]:

```
sns.boxplot(X_train_df['target'],X_train_df['sg'])
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d016648>
```



In [49]:

```
X_train_df['sg'].describe()
```

Out[49]:

```
count    256.000000
mean       1.017871
std        0.005518
min        1.005000
25%        1.015000
50%        1.020000
75%        1.020000
max        1.025000
Name: sg, dtype: float64
```

Checked on the internet that specific gravity of urine is an important factor in determining the chronic renal failure

1.4 Analysing feature AL(Albumin)

In [50]:

```
X_train_df['al'].describe()
```

Out[50]:

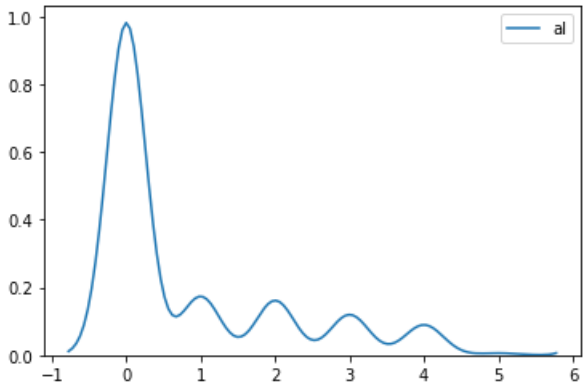
```
count    256.000000
mean       0.812500
std        1.272638
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        5.000000
Name: al, dtype: float64
```

In [51]:

```
sns.kdeplot(X_train_df['al'])
```

Out[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d043388>
```



It shows that distribution of albumin is right skewed.
Lesser number of people have high albumin levels

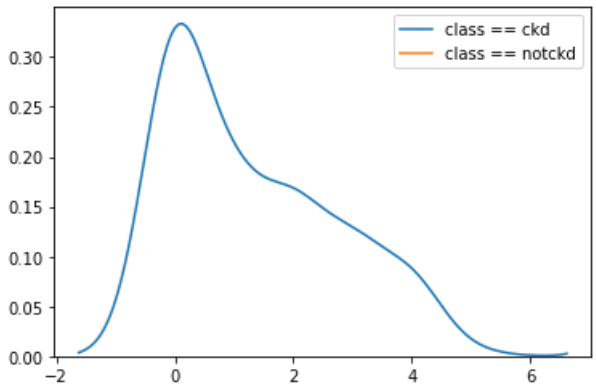
In [52]:

```
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 1,'al'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 0,'al'], label = 'class == notckd')
```

```
C:\Users\capiot\anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must h
ave variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
```

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d0c4b08>
```

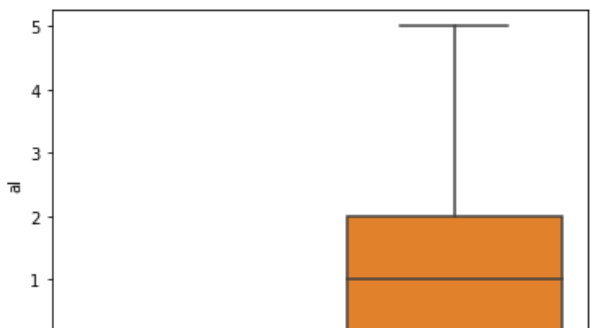


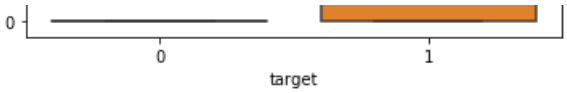
In [53]:

```
sns.boxplot(x=X_train_df['target'],y=X_train_df['al'])
```

Out[53]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d196cc8>
```





As seen from the above kde plot and box plot only three values of albumin contribute to 'notckd'.
So let's form two groups based on kde.

In [54]:

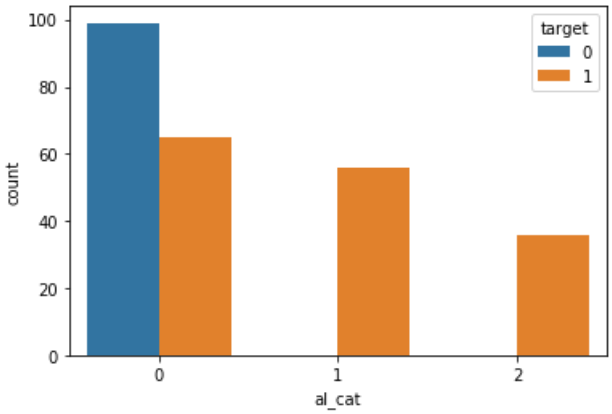
```
def formAlBins(x):
  if x == 0:
    return 0
  elif x >=1 and x <=2:
    return 1
  else:
    return 2

feature_df['al_cat'] = X_train_df['al'].apply(lambda x : formAlBins(x))
```

In [55]:

```
sns.countplot(x='al_cat',hue='target',data=feature_df)
```

Out[55]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d208948>
```



In [56]:

```
feature_df[['al_cat','target']].corr()
```

Out[56]:

	al_cat	target
al_cat	1.000000	0.544739
target	0.544739	1.000000

Forming bins has increased the correlation between albumin and target from 0.50 to 0.54

1.5 Analysing feature su(Sugar)

In [57]:

```
X_train_df['su'].value_counts()
```

Out[57]:

```
0.0    222
1.0     10
2.0      8
```

```
2.0       8
3.0       8
4.0       6
5.0       2
Name: su, dtype: int64
```

In [58]:

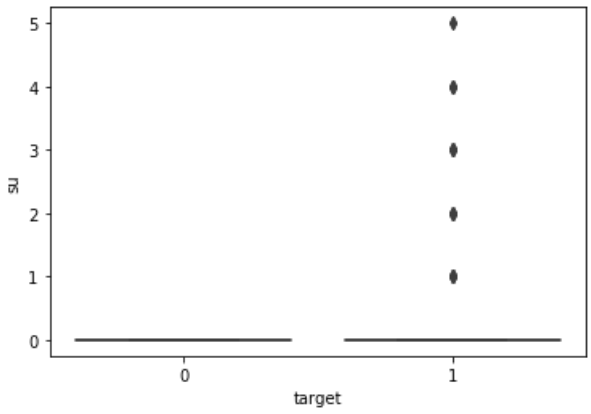
```
# Facing error while running this cell
#sns.kdeplot(X_train_df.loc[X_train_df['target'] == 1,'su'],label='class=ckd')
#sns.kdeplot(X_train_df.loc[X_train_df['target'] == 0,'su'],label='class=notckd')
```

In [59]:

```
sns.boxplot(X_train_df['target'],X_train_df['su'])
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d270608>
```



In [60]:

```
X_train_df['su'].value_counts()
```

Out[60]:

```
0.0    222
1.0     10
2.0      8
3.0      8
4.0      6
5.0      2
Name: su, dtype: int64
```

Form two bins :
su - 0
su - 1

In [61]:

```
feature_df['su_bin'] = X_train_df['su'].apply(lambda x : 0 if x==0 else 1)
```

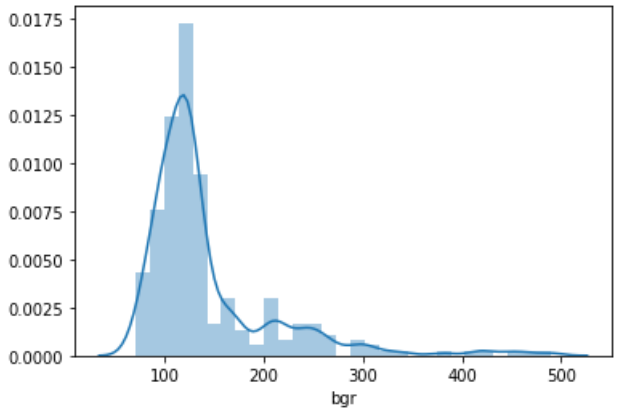
1.6 Analysis of feature BGR(Blood Glucose Random)

In [62]:

```
sns.distplot(X_train_df['bgr'],label='distribution plot of bgr')
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d2a5048>
```



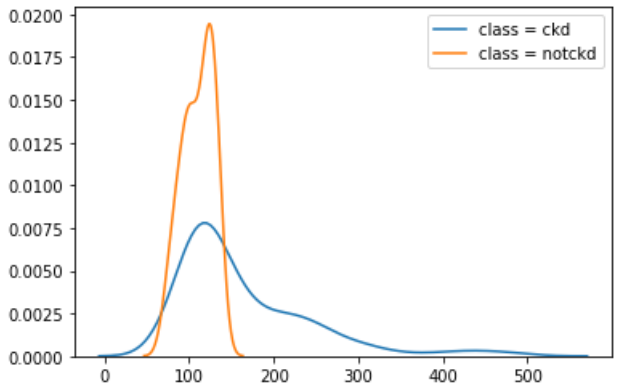
It's right skewed

In [63]:

```
sns.kdeplot(X_train_df.loc[X_train_df['target']==1,'bgr'],label='class = ckd')
sns.kdeplot(X_train_df.loc[X_train_df['target']==0,'bgr'],label='class = notckd')
```

Out[63]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d391ac8>
```



In [64]:

```
q = np.percentile(X_train_df[~X_train_df['bgr'].isnull() & (X_train_df['target'] == 0)]['bgr'],q=[75,90,99])
q
```

Out[64]:

```
array([125.  , 131.2 , 139.02])
```

So in the above piece of code,I am trying to find out the quantiles for 'blood group random' with target = 0("Not Ckd")
Since we can see lower value of blood group random to not have ckd, we can clearly form two groups based on tha above observation.
One for the "notCkd" class and other for ckd class

In [65]:

```
feature_df['bgr_bin'] = X_train_df['bgr'].apply(lambda x : 0 if x <= 140 else 1)
```

In [66]:

```
abs(feature_df[['bgr_bin','target']].corr())
```

Out[66]:

	bgr_bin	target

	bgr_bin	target
bgr_bin	1.00000	0.48236
target	0.48236	1.00000

Improved bgr's correlation value from 0.41 to 0.482

In [67]:

```
feature_df.head()
```

Out[67]:

	target	age_bins	bp_bins	age_bp_mean	age_bp_max	age_bp_min	age_bp_count	al_cat	su_bin	bgr_bin
179	1	3	2	76.470588	100.0	60.0	85.0	1	0	0
270	0	1	1	72.790698	100.0	60.0	43.0	0	0	0
323	0	2	1	78.750000	180.0	50.0	112.0	0	0	0
399	0	2	1	78.750000	180.0	50.0	112.0	0	0	0
40	1	2	2	78.750000	180.0	50.0	112.0	1	0	0

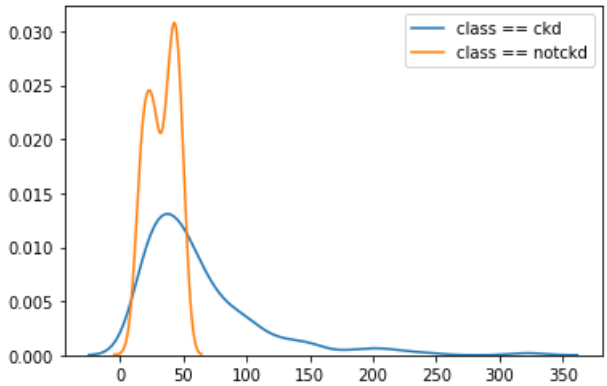
1.7 Analysing feature bu

In [68]:

```
## Let's look at feature bu
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 1,'bu'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 0,'bu'], label = 'class == notckd')
```

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d39adc8>
```



In [69]:

```
from scipy.stats import spearmanr
spearmanr(X_train_df['bu'],X_train_df['target'])
```

Out[69]:

```
SpearmanrResult(correlation=0.38072603708044706, pvalue=2.9632239551884964e-10)
```

In [70]:

```
X_train_df[['bu','target']].corr()
```

Out[70]:

	bu	target
bu	1.000000	0.352857

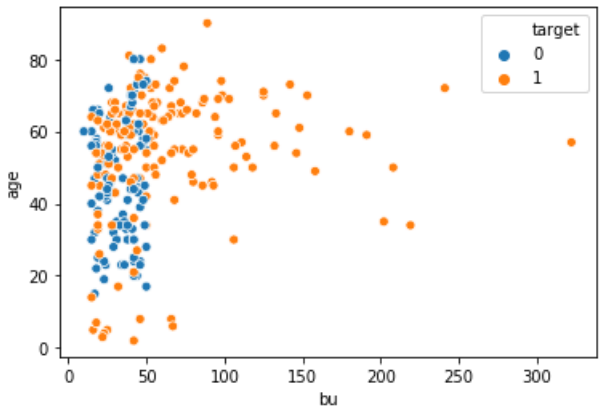
	bu	target
bu	1.000000	0.353857
target	0.353857	1.000000

In [71]:

```
sns.scatterplot('bu','age',hue = 'target',data = X_train_df)
```

Out[71]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d4b5d88>
```



With blood urea values less than 50,number of CKD cases are comparitively less as compared to NotCKD cases.
BloodUrea along with Age doesn't prove to be useful in forming binary features or any sort of features.

In [72]:

```
## Coming up with the border value for creating binary bins
q = np.percentile(X_train_df[X_train_df['target'] == 0]['bu'],q=[75,90,99])
print(q)
```

```
[43. 48. 50.]
```

In [73]:

```
feature_df['bu_bin'] = X_train_df['bu'].apply(lambda x : 0 if x <= 50 else 1)
feature_df['bu_bin'].value_counts()
```

Out[73]:

```
0    179
1     77
Name: bu_bin, dtype: int64
```

In [74]:

```
feature_df.head()
```

Out[74]:

	target	age_bins	bp_bins	age_bp_mean	age_bp_max	age_bp_min	age_bp_count	al_cat	su_bin	bgr_bin	bu_bin
179	1	3	2	76.470588	100.0	60.0	85.0	1	0	0	1
270	0	1	1	72.790698	100.0	60.0	43.0	0	0	0	0
323	0	2	1	78.750000	180.0	50.0	112.0	0	0	0	0
399	0	2	1	78.750000	180.0	50.0	112.0	0	0	0	0
40	1	2	2	78.750000	180.0	50.0	112.0	1	0	0	1

In [75]:

```
feature_df[['bu_bin','target']].corr()['bu_bin']
```

Out[75]:

```
bu_bin    1.000000
target    0.520819
Name: bu_bin, dtype: float64
```

In [76]:

```
spearmanr(feature_df['bu_bin'],feature_df['target'])
```

Out[76]:

```
SpearmanrResult(correlation=0.5208187587660582, pvalue=3.3517945961596853e-19)
```

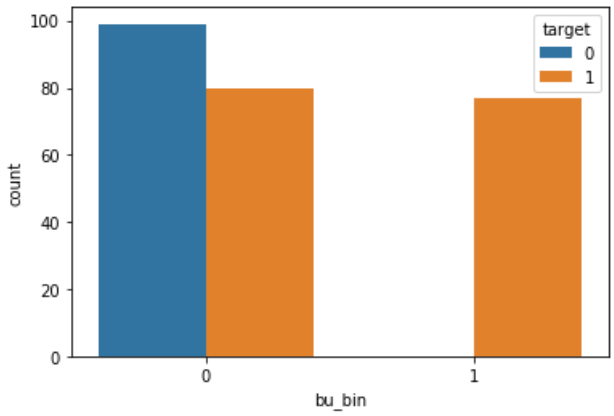
p-value is less than significance level and therefore the null hypothesis that attributes are independent is rejected and the alternate hypothesis that there is some correlation between the two attributes is not rejected.
Also the correlation coefficient has been increased significantly after forming binary features with bu

In [77]:

```
sns.countplot(x='bu_bin',hue='target',data=feature_df)
```

Out[77]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d99b248>
```



In [78]:

```
X_train_df.select_dtypes(include=['float64','int64']).columns
```

Out[78]:

```
Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'bgr', 'bu', 'sc',
       'sod', 'pot', 'hemo', 'rc', 'target'],
      dtype='object')
```

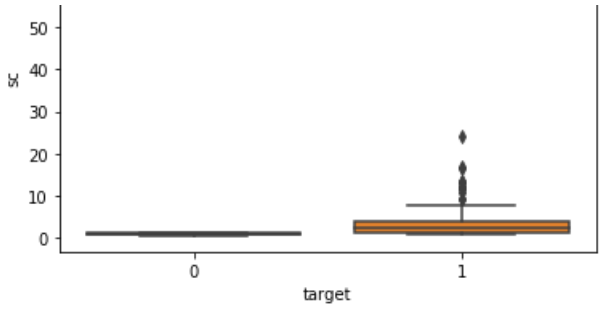
1.8 Analysing sc - Serum Creatinine

In [79]:

```
sns.boxplot('target','sc',data = X_train_df)
```

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407d9e9248>
```

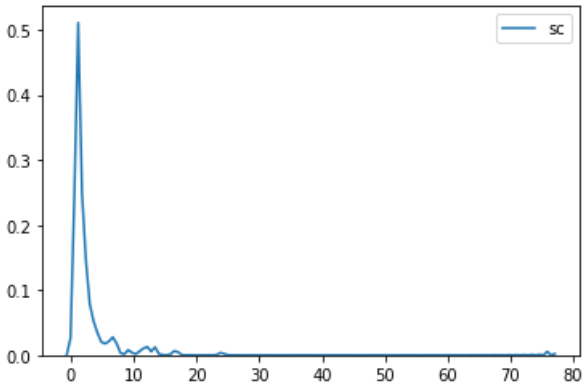


In [80]:

```
sns.kdeplot(X_train_df['sc'])
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407da31b48>
```



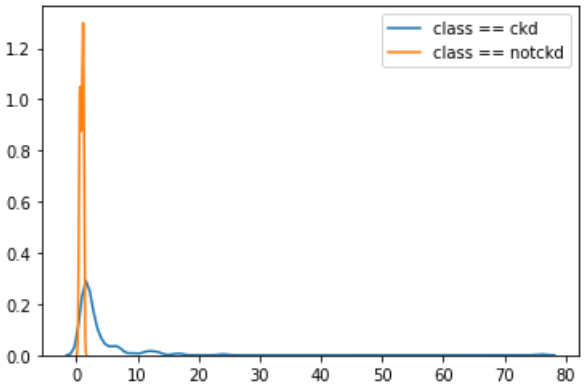
The distribution is right skewed and contains outliers.

In [81]:

```
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'ckd','sc'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'notckd','sc'], label = 'class == notckd
')
```

Out[81]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407da5c648>
```



Normal creatinine levels are in the range of 0.5 to 1.2 mg/dL
As can be seen from the plot above, those in the normal range have less chances of ckd as compared to those beyond the normal range.

In [82]:

```
qSc = np.percentile(X_train_df.loc[X_train_df['classification'] == 'notckd','sc'],q=[75,90,99])
```

```
qSc
```

Out[82]:

```
array([1.1, 1.2, 1.3])
```

Even the quantile values suggest the same.
Let's form binary bins with 0 for normal range of blood creatinine values and 1 for values beyond the normal range.

In [83]:

```
feature_df['sc_bin'] = X_train_df['sc'].apply(lambda x : 0 if x <= 1.2 else 1)
```

In [84]:

```
feature_df[['sc_bin','target']].corr()['target']
```

Out[84]:

```
sc_bin    0.751925
target    1.000000
Name: target, dtype: float64
```

In [85]:

```
X_train_df[['sc','target']].corr()['target']
```

Out[85]:

```
sc        0.258217
target    1.000000
Name: target, dtype: float64
```

Correlation value has been increased from 0.25 to 0.75

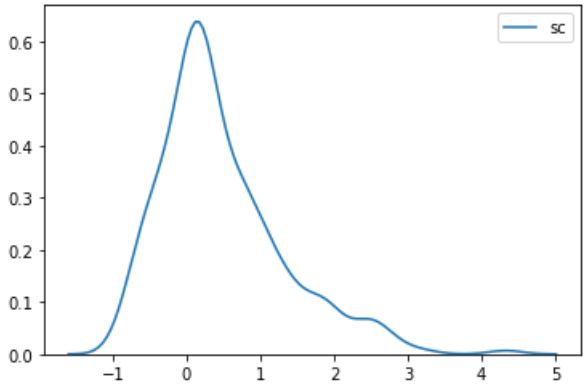
Let's try to normalize this right skewed feature

In [86]:

```
log_norm = np.log(X_train_df['sc'])
sns.kdeplot(log_norm)
```

Out[86]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407dadc9c8>
```



In [87]:

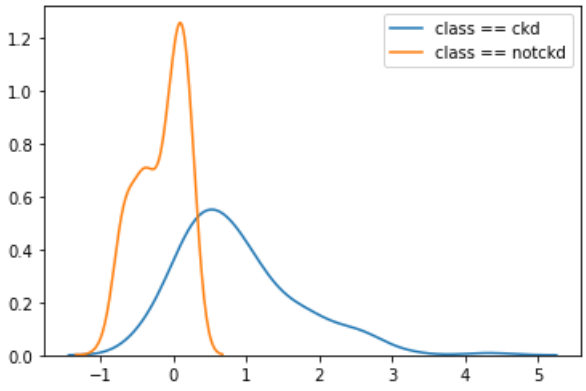
```
feature_df['log_norm_sc'] = log_norm
```

In [88]:

```
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'log_norm_sc'], label = 'class == ckd')
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'log_norm_sc'], label = 'class == notckd')
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407db89848>
```



Let us keep it like this.
Not forming any binary feature since we already have built them earlier.

In [89]:

```
feature_df[['log_norm_sc','target']].corr()
```

Out[89]:

	log_norm_sc	target
log_norm_sc	1.000000	0.605407
target	0.605407	1.000000

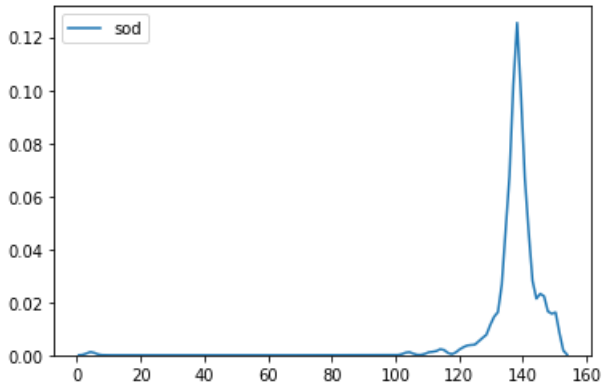
1.9 Analysing sod - Sodium

In [90]:

```
sns.kdeplot(X_train_df['sod'])
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407dbc7048>
```

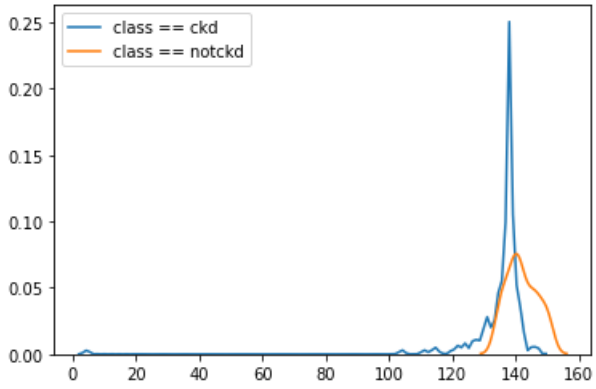


In [91]:

```
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'ckd','sod'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'notckd','sod'], label = 'class == notck
d')
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407dc81108>
```



In [92]:

```
qSod = np.percentile(X_train_df.loc[X_train_df['classification'] == 'notckd','sod'],q=[0,25,50,75,99])
qSod
```

Out[92]:

```
array([135., 138., 141., 146., 150.])
```

In [93]:

```
qSod = np.percentile(X_train_df.loc[X_train_df['classification'] == 'ckd','sod'],q=[0,25,50,75,99])
qSod
```

Out[93]:

```
array([  4.5 , 135.  , 138.  , 138.  , 145.44])
```

Normal range of Sodium lies in between 135-145 mEq/L
Let's create binary feature.

In [94]:

```
feature_df['sod_bin'] = X_train_df['sod'].apply(lambda x : 1 if x <= 138 else 0)
feature_df[['sod_bin','target']].corr()
```

Out[94]:

	sod_bin	target
sod_bin	1.00000	0.54975
target	0.54975	1.00000

In [95]:

```
X_train_df[['sod','target']].corr()
```

Out[95]:

	sod	target
sod	1.000000	-0.322329
target	-0.322329	1.000000

Correlation value has increased from 0.32 to 0.54

Let's normalize this feature

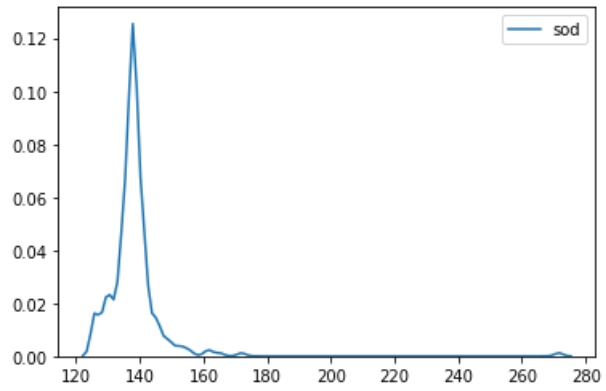
- let's see if we can improve correlation with target variable

In [96]:

```
median_sod = np.median(X_train_df['sod'])

def reflector(x):
  if x < median_sod:
    dev = median_sod-x
    return median_sod + dev
  elif x > median_sod:
    dev = x - median_sod
    return median_sod - dev
  else:
    return x
reflected_sod = X_train_df['sod'].apply(lambda x : reflector(x))
sns.kdeplot(reflected_sod)
```

Out[96]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407dcae4c8>
```

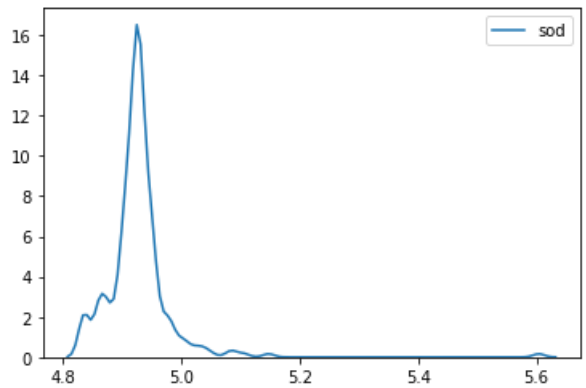


In [97]:

```
## Converting this rightskewed distribution to normal distribution
log_norm_sod = np.log(reflected_sod)
sns.kdeplot(log_norm_sod)
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407dd70588>
```

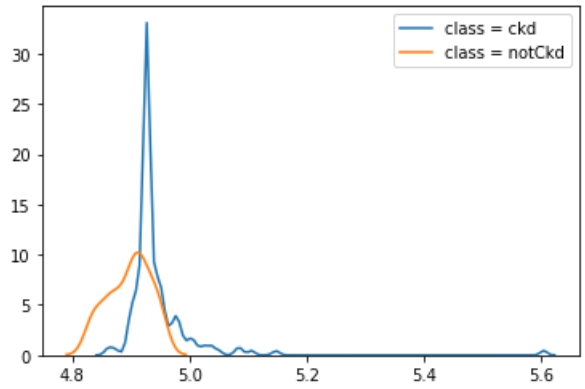


In [98]:

```
feature_df['log_norm_sod'] = log_norm_sod
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'log_norm_sod'],label="class = ckd")
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'log_norm_sod'],label="class = notCkd")
```

Out[98]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407ddddf48>
```



In [99]:

```
feature_df[['target','log_norm_sod']].corr()
```

Out[99]:

	target	log_norm_sod
target	1.00000	0.38379
log_norm_sod	0.38379	1.00000

In [100]:

```
qLogNormSodNCkd = np.percentile(feature_df[feature_df['target'] == 0]['log_norm_sod'],q=[0,25,50,75
,99])
qLogNormSodCkd = np.percentile(feature_df[feature_df['target'] == 1]['log_norm_sod'],q=[0,25,50,75,
99])

print(qLogNormSodNCkd)
print(qLogNormSodCkd)
```

```
[4.83628191 4.86753445 4.90527478 4.92725369 4.94875989]
[4.8598124  4.92725369 4.92725369 4.94875989 5.12422704]
```

75th percentile value for NCkd is same as 25th percentile value for CKD.
We'll use this mark as a boundary for creating our binary bins

In [101]:

```
feature_df['norm_sod_bin'] = feature_df['log_norm_sod'].apply(lambda x : 0 if x <= 4.92 else 1)
feature_df[['target','norm_sod_bin']].corr()
```

Out[101]:

	target	norm_sod_bin
target	1.00000	0.54975
norm_sod_bin	0.54975	1.00000

The correlation value still remains the same.
There was no point building this feature.

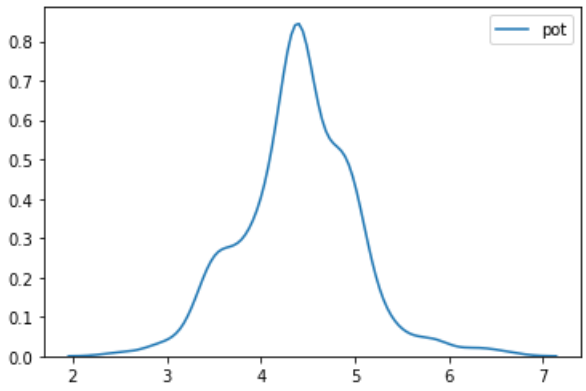
1.10 Analysing pot - Potassium

In [102]:

```
sns.kdeplot(X_train_df['pot'])
```

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407de4df88>
```

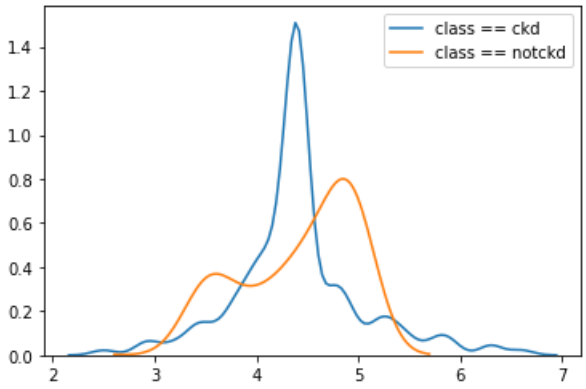


In [103]:

```
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'ckd','pot'], label = 'class == ckd')
sns.kdeplot(X_train_df.loc[X_train_df['classification'] == 'notckd','pot'], label = 'class == notck
d')
```

Out[103]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407ee96b48>
```



In [104]:

```
X_train_df[['target','pot']].corr()
```

Out[104]:

	target	pot
target	1.000000	-0.010682
pot	-0.010682	1.000000

I don't think we can do much with this attribute.

1.11 Analysing hemo - Hemoglobin

In [105]:

```
X_train_df[['target','hemo']].corr()
```

Out[105]:

Out[105]:

	target	hemo
target	1.00000	-0.73239
hemo	-0.73239	1.00000

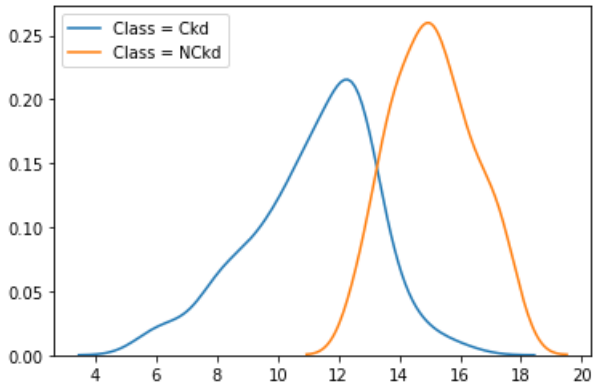
Hemoglobin shows negative correlation

In [106]:

```
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 1,'hemo'],label = "Class = Ckd")
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 0,'hemo'],label = "Class = NCkd")
```

Out[106]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407eecc348>
```



In [107]:

```
qPotNCkd = np.percentile(X_train_df[X_train_df['target'] == 1]['hemo'],q=[0,25,50,75,99])
qPotCkd = np.percentile(X_train_df[X_train_df['target'] == 0]['hemo'],q=[0,25,50,75,99])

print(qPotNCkd)
print(qPotCkd)
```

```
[ 5.8    9.8   11.4   12.65  15.264]
[12.65 14.   15.   16.05 17.8 ]
```

In [108]:

```
feature_df['hemo_bin'] = X_train_df['hemo'].apply(lambda x : 0 if x <= 12.65 else 1)
feature_df[['target','hemo_bin']].corr()
```

Out[108]:

	target	hemo_bin
target	1.000000	-0.798655
hemo_bin	-0.798655	1.000000

Forming bins has increased the correlation.
Also By forming bins we are trying to generalize to some extent.

1.12 Analysing RC

In [109]:

```
X_train_df[['target','rc']].corr()
```

Out[109]:

Out[109]:

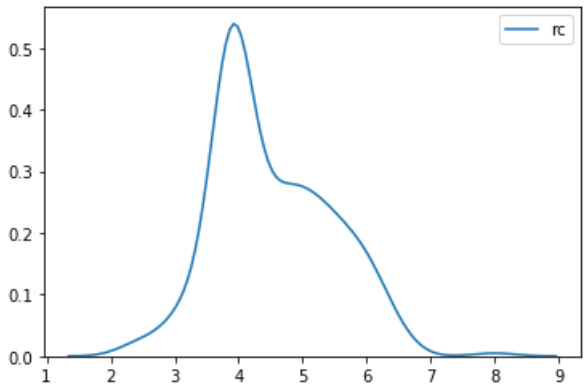
	target	rc
target	1.00000	-0.73315
rc	-0.73315	1.00000

In [110]:

```
sns.kdeplot(X_train_df['rc'])
```

Out[110]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407efe3d88>
```

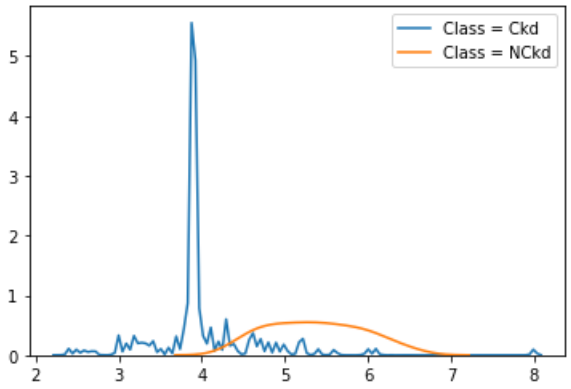


In [111]:

```
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 1,'rc'],label = "Class = Ckd")
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 0,'rc'],label = "Class = NCkd")
```

Out[111]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407f007288>
```



In [112]:

```
qRcNCkd = np.percentile(X_train_df[X_train_df['target'] == 1]['rc'],q=[0,25,50,75,99])
qRcCkd = np.percentile(X_train_df[X_train_df['target'] == 0]['rc'],q=[0,25,50,75,99])

print(qRcNCkd)
print(qRcCkd)
```

```
[2.3   3.9   3.9   4.    6.044]
[4.4 4.9 5.3 5.8 6.5]
```

In [113]:

```
feature_df['rc_bin'] = X_train_df['rc'].apply(lambda x : 0 if x <= 4.4 else 1)
feature_df[['target','rc_bin']].corr()
```

Out[113]:

	target	rc_bin
target	1.000000	-0.822702
rc_bin	-0.822702	1.000000

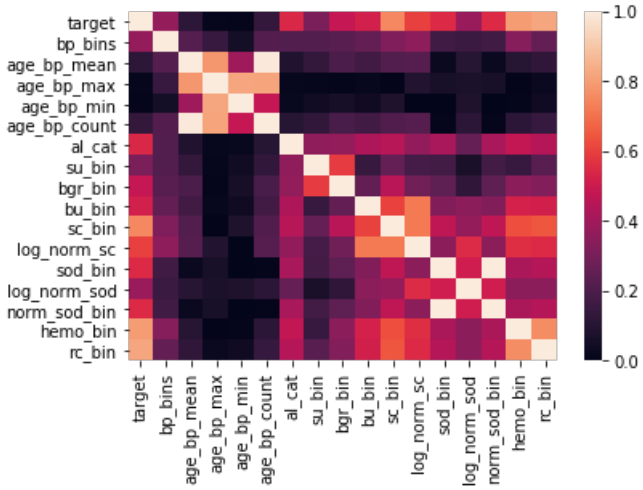
4.4 seems to be a good split point

In [114]:

```
sns.heatmap(abs(feature_df.corr()))
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407f035d48>
```



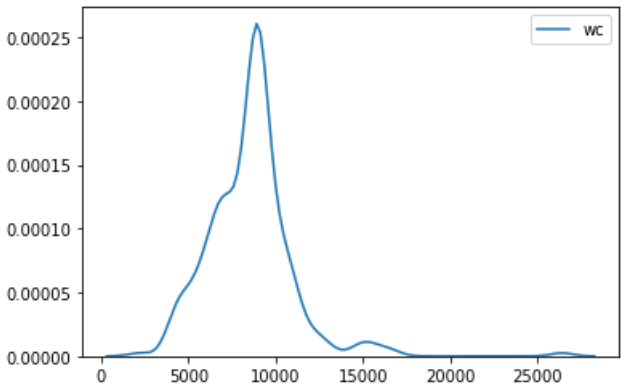
1.13 Analysing WC(White Blood Cell count)

In [115]:

```
sns.kdeplot(X_train_df['wc'])
```

Out[115]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407878d288>
```



In [116]:

```
X_train_df[['wc','target']].corr()
```

Out[116]:

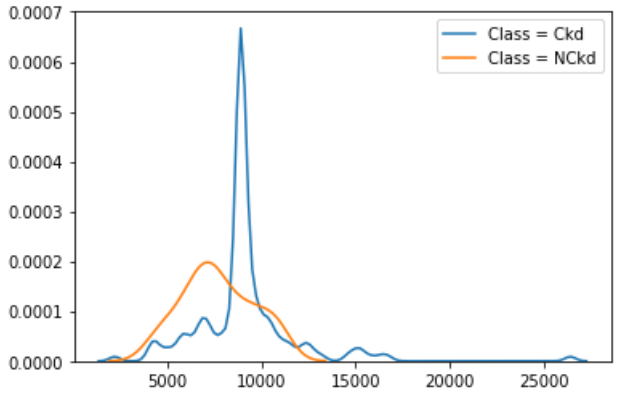
	wc	target
wc	1.000000	0.279009
target	0.279009	1.000000

In [117]:

```
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 1,'wc'],label = "Class = Ckd")
sns.kdeplot(X_train_df.loc[X_train_df['target'] == 0,'wc'],label = "Class = NCkd")
```

Out[117]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407eee1a48>
```



In [118]:

```
 qWcNCkd = np.percentile(X_train_df[X_train_df['target'] == 1]['wc'],q=[0,25,50,75,99])
qWcCkd = np.percentile(X_train_df[X_train_df['target'] == 0]['wc'],q=[0,25,50,75,99])
print(qWcNCkd)
print(qWcCkd)
```

```
[ 2200.  8600.  8900.  9600. 16476.]
[ 4300.  6350.  7500.  9200. 11000.]
```

In [119]:

```
feature_df['wc_bin'] = X_train_df['wc'].apply(lambda x : 0 if x <=8600 else 1)
feature_df[['target','wc_bin']].corr()
```

Out[119]:

	target	wc_bin
target	1.000000	0.444954
wc_bin	0.444954	1.000000

Dropping GroupBy features

In [120]:

```
feature_df.head()
```

Out[120]:

	target	age_bins	bp_bins	age_bp_mean	age_bp_max	age_bp_min	age_bp_count	al_cat	su_bin	bgr_bin	bu_bin	sc_bin	log_
179	1	3	2	76.470588	100.0	60.0	85.0	1	0	0	1	1	
270	0	1	1	72.790698	100.0	60.0	43.0	0	0	0	0	0	
323	0	2	1	78.750000	180.0	50.0	112.0	0	0	0	0	0	

	target	age_bins	bp_bins	age_bp_mean	age_bp_max	age_bp_min	age_bp_count	al_cat	su_bin	bgr_bin	bu_bin	sc_bin	log_
399	0	2	1	78.750000	180.0	50.0	112.0	0	0	0	0	0	
40	1	2	2	78.750000	180.0	50.0	112.0	1	0	0	1	1	

Heatmap suggests that groupby features formed by age and bp are practically of no use.
So we'll drop this features.

In [121]:

```
feature_df.drop(['age_bp_mean','age_bp_max','age_bp_min','age_bp_count'],axis = 1,inplace = True)
feature_df.head(4)
```

Out[121]:

	target	age_bins	bp_bins	al_cat	su_bin	bgr_bin	bu_bin	sc_bin	log_norm_sc	sod_bin	log_norm_sod	norm_sod_bin	hemo_
179	1	3	2	1	0	0	1	1	0.832909	1	4.927254	1	
270	0	1	1	0	0	0	0	0	0.095310	0	4.875197	0	
323	0	2	1	0	0	0	0	0	0.095310	0	4.890349	0	
399	0	2	1	0	0	0	0	0	0.095310	0	4.905275	0	

In [122]:

```
print("Number of newly discovered features with Univariate Analysis : ",len(feature_df.columns)-1)
```

```
Number of newly discovered features with Univariate Analysis :  14
```

Multivariate Feature Analysis

In [123]:

```
numeric_df = X_train_df.select_dtypes(include=['int64','float64'])
```

Since the number of features is more and pairplots will form a big grid of scatter plot's and kde which will be difficult to look at.
Let's see which attributes should be seen together to get some idea about feature interaction.

In [124]:

```
numeric_df.columns
```

Out[124]:

```
Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'bgr', 'bu', 'sc',
       'sod', 'pot', 'hemo', 'rc', 'target'],
      dtype='object')
```

In [125]:

```
cat_df = X_train_df.select_dtypes(include=['object'])
cat_df.columns
```

Out[125]:

```
Index(['pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane',
       'classification'],
      dtype='object')
```

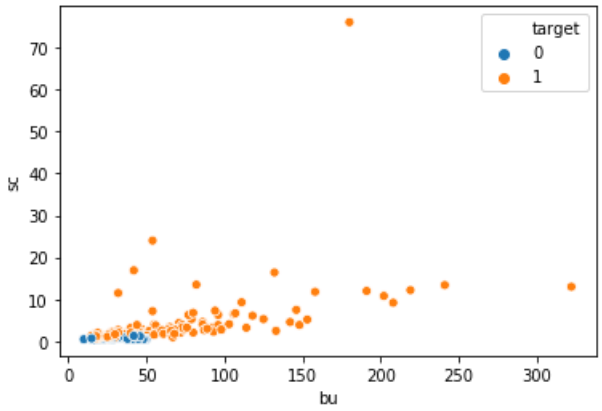
It has been observed that there is certain permissible level of sc(serum creatinine) and bu(blood urea) beyond which there are chances of a person contracting chronic kidney disease.
https://www.kidney.org/atoz/content/kidneytests

In [126]:

```
sns.scatterplot(x = 'bu',y='sc',hue='target',data=X_train_df)
```

Out[126]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407f24d288>
```



Looking at the simple scatter plot we can form a simple binary feature.

In [127]:

```
def sc_bu_bin(x):
    if x['sc'] <= 1.4 and x['bu'] <= 50:
        return 0
    else:
        return 1
feature_df['sc_bu_bin'] = X_train_df.apply(lambda x : sc_bu_bin(x),axis = 1)
feature_df[['sc_bu_bin','target']].corr()
```

Out[127]:

	sc_bu_bin	target
sc_bu_bin	1.000000	0.694777
target	0.694777	1.000000

In [128]:

```
feature_df.head()
```

Out[128]:

	target	age_bins	bp_bins	al_cat	su_bin	bgr_bin	bu_bin	sc_bin	log_norm_sc	sod_bin	log_norm_sod	norm_sod_bin	hemo_
179	1	3	2	1	0	0	1	1	0.832909	1	4.927254	1	
270	0	1	1	0	0	0	0	0	0.095310	0	4.875197	0	
323	0	2	1	0	0	0	0	0	0.095310	0	4.890349	0	
399	0	2	1	0	0	0	0	0	0.095310	0	4.905275	0	
40	1	2	2	1	0	0	1	1	0.741937	1	4.927254	1	

In [129]:

```
feature_df['acr'] = X_train_df['al']/X_train_df['sc']
feature_df[['acr','target']].corr()
```

Out[129]:

	acr	target
acr	1.000000	0.417466
target	0.417466	1.000000

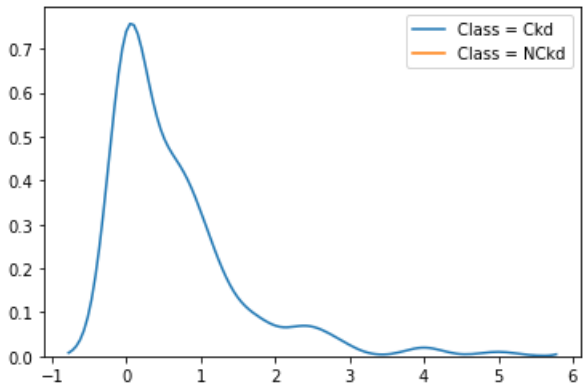
In [130]:

```
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'acr'],label = "Class = Ckd")
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'acr'],label = "Class = NCkd")
```

```
C:\Users\capiot\anaconda3\lib\site-packages\seaborn\distributions.py:288: UserWarning: Data must h
ave variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
```

Out[130]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407f344688>
```



Since we know Albumin,Creatinine and Blood Urea are important in deciding CKD.
Let's form Polynomial features and see if we can get anything.

In [131]:

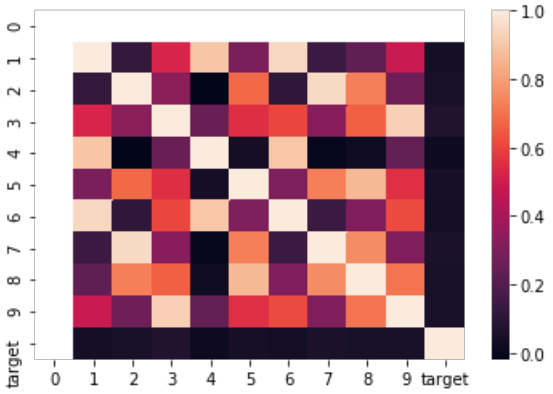
```
from sklearn.preprocessing import PolynomialFeatures
polyFeatures = PolynomialFeatures(2)
imp_feat_df = polyFeatures.fit_transform(X_train_df[['sc','al','bu']])
```

In [132]:

```
impt_feat_df = pd.DataFrame(imp_feat_df)
impt_feat_df['target'] = feature_df['target']
sns.heatmap(impt_feat_df.corr())
```

Out[132]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407f3ac088>
```



All developed polynomial features have almost 0 correlation with target variable.

Applying Box-Cox Transformations to continuous attributes

Let's apply Box-Cox tranform to all of the numeric features and get new normally distributed features.
Take each of the sample from train data and get it's probability distribution function.
I am talking about Multivariate_normal.pdf and see if we can build a binary feature.

Box-Cox Transformations take only positive data.
So normalizing this data using min-max scaler.

In [133]:

```
from sklearn.preprocessing import MinMaxScaler
numeric_x_df = numeric_df.drop(['id','target'],axis = 1)
numeric_y_df = numeric_df['target']

scaler = MinMaxScaler()
numeric_x_transformed = scaler.fit_transform(numeric_x_df)
numeric_x_trans_df = pd.DataFrame(numeric_x_transformed,columns = numeric_x_df.columns)
```

In [134]:

```
from scipy.stats import boxcox

# Forming a dataFrame to contain box-cox transformedd variables
box_cox_df = pd.DataFrame()

#Transforming all numeric columns except id and target
for col in numeric_x_trans_df.columns:
    if col != 'id' and col != 'target':
        box_cox_df['box_cox_' + col] = boxcox(numeric_x_trans_df[col]+1)[0]
```

In [135]:

```
box_cox_df.columns
```

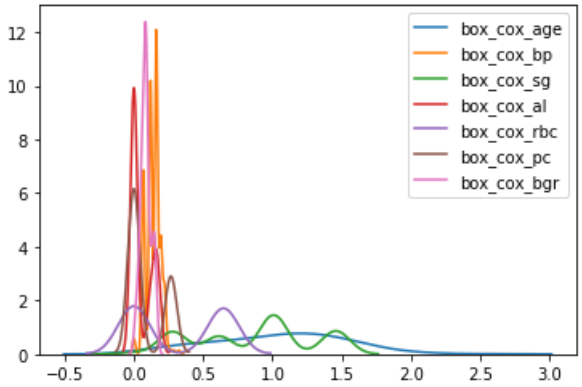
Out[135]:

```
Index(['box_cox_age', 'box_cox_bp', 'box_cox_sg', 'box_cox_al', 'box_cox_su',
       'box_cox_rbc', 'box_cox_pc', 'box_cox_bgr', 'box_cox_bu', 'box_cox_sc',
       'box_cox_sod', 'box_cox_pot', 'box_cox_hemo', 'box_cox_rc'],
      dtype='object')
```

In [136]:

```
for col in box_cox_df.columns[:8]:
    try:
        sns.kdeplot(box_cox_df[col])
    except:
        print(col)
```

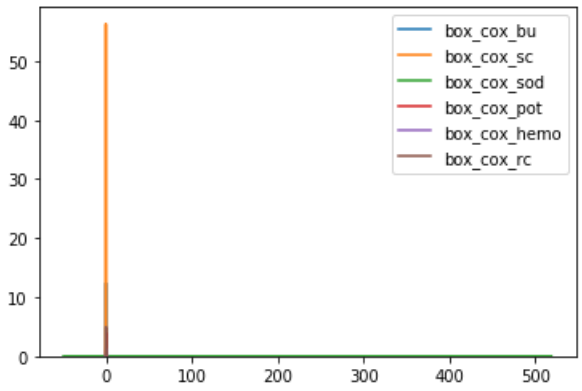
box_cox_su



In [137]:

```
for col in box_cox_df.columns[8:]:
```

```
    try:
        sns.kdeplot(box_cox_df[col])
    except:
        print(col)
```



The Distributions are decently normal.
Actually they aren't! Hahaha!

In [138]:

```
from sklearn.preprocessing import  StandardScaler

scaler  = StandardScaler().fit(box_cox_df)
box_cox_std = scaler.transform(box_cox_df)

## Let's calculate covariance and mean for multivariate analysis
sigma = np.cov(box_cox_std.T)
print("Shape of Covariance matrix : ",sigma.shape)
mean = np.mean(box_cox_std,axis = 0)
print("Mean of box cox_df : ",mean)
```

```
Shape of Covariance matrix :  (14, 14)
Mean of box cox_df :  [-3.78169718e-16 -4.85722573e-17  3.33066907e-16  5.89805982e-17
  6.24500451e-17 -6.93889390e-18  9.02056208e-17  1.80411242e-16
  1.42247325e-16  1.17961196e-16  7.60676244e-16 -1.63064007e-16
 -2.35922393e-16 -4.44089210e-16]
```

Calculating multivariate normal pdf

In [139]:

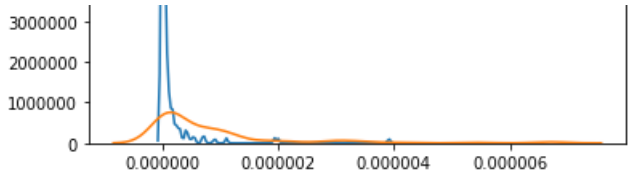
```
from scipy.stats import multivariate_normal
multivariate_pdf = multivariate_normal.pdf(box_cox_std,mean=mean,cov=sigma)
feature_df['multivariate_pdf'] = multivariate_pdf
feature_df['log_multivariate_pdf'] = np.log(multivariate_pdf)
```

In [140]:

```
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'multivariate_pdf'], label = 'class == ckd')
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'multivariate_pdf'], label = 'class == notckd'
)
```

Out[140]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407f4f7c88>
```



In [141]:

```
feature_df[['multivariate_pdf','target']].corr()
```

Out[141]:

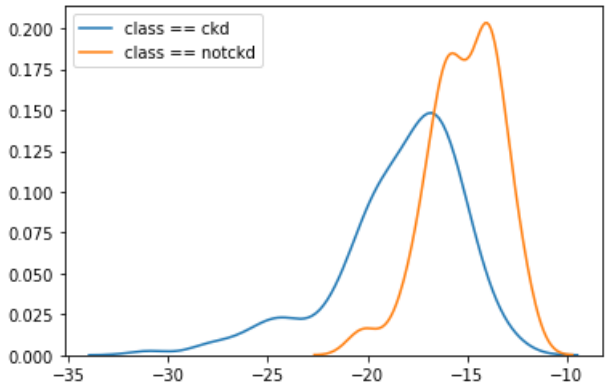
	multivariate_pdf	target
multivariate_pdf	1.000000	-0.374873
target	-0.374873	1.000000

In [142]:

```
sns.kdeplot(feature_df.loc[feature_df['target'] == 1,'log_multivariate_pdf'], label = 'class == ckd
')
sns.kdeplot(feature_df.loc[feature_df['target'] == 0,'log_multivariate_pdf'], label = 'class == not
ckd')
```

Out[142]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2407f5c0dc8>
```



In [143]:

```
feature_df[['log_multivariate_pdf','target']].corr()
```

Out[143]:

	log_multivariate_pdf	target
log_multivariate_pdf	1.000000	-0.500745
target	-0.500745	1.000000

Let's try to form a binary feature using the above pdf

In [144]:

```
qLogMultiPdfNCkd = np.percentile(feature_df[feature_df['target'] == 0]['log_multivariate_pdf'],q=[0
,25,50,75,99])
qLogMultiPdfCkd = np.percentile(feature_df[feature_df['target'] == 1]['log_multivariate_pdf'],q=[0,
25,50,75,99])

print(qLogMultiPdfCkd)
print(qLogMultiPdfNCkd)
```

```
[-30.98424697 -19.74149681 -17.73784374 -16.24877048 -13.13937376]
[-20.4805743  -16.15454415 -15.00272175 -13.82512002 -11.91900243]
```

In [145]:

```
feature_df['log_multi_pdf_bin'] = feature_df['log_multivariate_pdf'].apply(lambda x : 1 if x <= -16
.24 else 0)
feature_df[['target','log_multi_pdf_bin']].corr()
```

Out[145]:

	target	log_multi_pdf_bin
target	1.000000	0.517886
log_multi_pdf_bin	0.517886	1.000000

Doesn't seem to be that useful.
But let's keep it.

In [146]:

```
feature_df.columns
```

Out[146]:

```
Index(['target', 'age_bins', 'bp_bins', 'al_cat', 'su_bin', 'bgr_bin',
       'bu_bin', 'sc_bin', 'log_norm_sc', 'sod_bin', 'log_norm_sod',
       'norm_sod_bin', 'hemo_bin', 'rc_bin', 'wc_bin', 'sc_bu_bin', 'acr',
       'multivariate_pdf', 'log_multivariate_pdf', 'log_multi_pdf_bin'],
      dtype='object')
```

Let's dump the generated features for future use

In [147]:

```
sampleFile = open('feature_df','wb')
pickle.dump(feature_df,sampleFile)
sampleFile.close()
```

In [148]:

```
feature_df.shape
```

Out[148]:

```
(256, 20)
```

In [ ]:

```

```