Imputation of missing values in numeric and categorical variables

January 31, 2021

[1]:
```
#import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
#import the datasets
df= pd.read_csv('hepatitis.csv')
```

[3]:
```
#print head
df.head()
```

[3]:
```
   age     sex steroid  antivirals fatigue malaise anorexia liver_big  \
0   30    male   False       False   False   False    False     False
1   50  female   False       False    True   False    False     False
2   78  female    True       False    True   False    False      True
3   31  female     NaN        True   False   False    False      True
4   34  female    True       False   False   False    False      True

  liver_firm spleen_palpable spiders ascites varices  bilirubin  \
0      False           False   False   False   False        1.0
1      False           False   False   False   False        0.9
2      False           False   False   False   False        0.7
3      False           False   False   False   False        0.7
4      False           False   False   False   False        1.0

   alk_phosphate   sgot  albumin  protime  histology class
0           85.0   18.0      4.0      NaN      False  live
1          135.0   42.0      3.5      NaN      False  live
2           96.0   32.0      4.0      NaN      False  live
3           46.0   52.0      4.0     80.0      False  live
4            NaN  200.0      4.0      NaN      False  live
```

[10]:
```
df.shape
```

[10]:
```
(155, 20)
```

```
[9]: df.isnull().sum()
```

```
[9]: age                 0
     sex                 0
     steroid             1
     antivirals          0
     fatigue             1
     malaise             1
     anorexia            1
     liver_big          10
     liver_firm         11
     spleen_palpable     5
     spiders             5
     ascites             5
     varices             5
     bilirubin           6
     alk_phosphate      29
     sgot                4
     albumin            16
     protime            67
     histology           0
     class               0
     dtype: int64
```

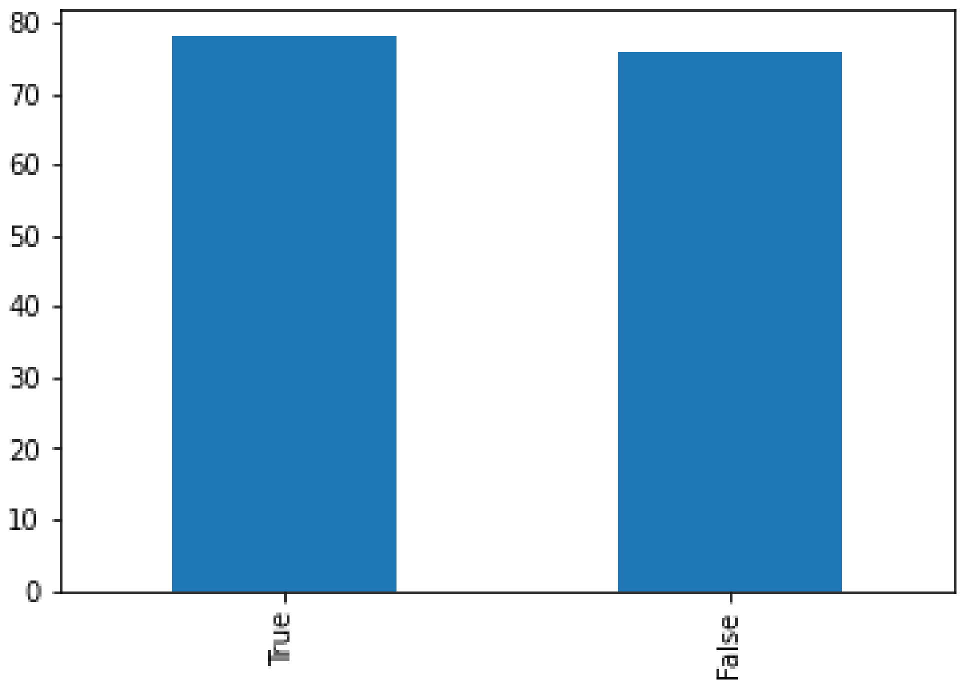
```
[16]: missing_catagorical=[var for var in df.columns if df[var].isnull().mean()>0 and
                          df[var].dtypes=='O']
```

```
[17]: missing_catagorical
```

```
[17]: ['steroid',
       'fatigue',
       'malaise',
       'anorexia',
       'liver_big',
       'liver_firm',
       'spleen_palpable',
       'spiders',
       'ascites',
       'varices']
```

```
[18]: df['steroid'].value_counts().plot.bar()
```

```
[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1e237990cd0>
```



```
[19]: df['steroid'].mode()
```

```
[19]: 0    True
      dtype: object
```

```
[25]: #filling the missing values NaN by mode
      df['steroid'].fillna('True', inplace=True)
      df['steroid'].isnull().sum()
```

```
[25]: 0
```

```
[ ]: #Likewise filling the missing values in catagorical variables by mode and␣
      ↪numerical variables by mean
     # as per the rule if columns contains 20% values as NaN values, we can drop but␣
      ↪I am here replacing all by mode
```

```
[26]: cols=['fatigue',
       'malaise',
       'anorexia',
       'liver_big',
       'liver_firm',
       'spleen_palpable',
       'spiders',
       'ascites',
       'varices']
```

[39]:
```
df['fatigue'].mode()
```

[39]:
```
0    True
dtype: object
```

[40]:
```
df['fatigue'].fillna('True', inplace=True)
df['fatigue'].isnull().sum()
```

[40]:
```
0
```

[41]:
```
# A good technique to replace the NaN values in catagorical variables is␣
 ↪replaced by Missing Keyword
df['anorexia'].fillna('Missing', inplace=True)
```

[42]:
```
df['anorexia'].isnull().sum()
```

[42]:
```
0
```

[43]:
```
df['liver_big'].fillna('Missing', inplace=True)
df['liver_firm'].fillna('Missing', inplace=True)
df['spleen_palpable'].fillna('Missing', inplace=True)
df['spiders'].fillna('Missing', inplace=True)
df['ascites'].fillna('Missing', inplace=True)
df['varices'].fillna('Missing', inplace=True)
```

[44]:
```
df.isnull().sum()
```

[44]:
```
age                 0
sex                 0
steroid             0
antivirals          0
fatigue             0
malaise             1
anorexia            0
liver_big           0
liver_firm          0
spleen_palpable     0
spiders             0
ascites             0
varices             0
bilirubin           6
alk_phosphate      29
sgot                4
albumin            16
protime            67
histology           0
class               0
dtype: int64
```

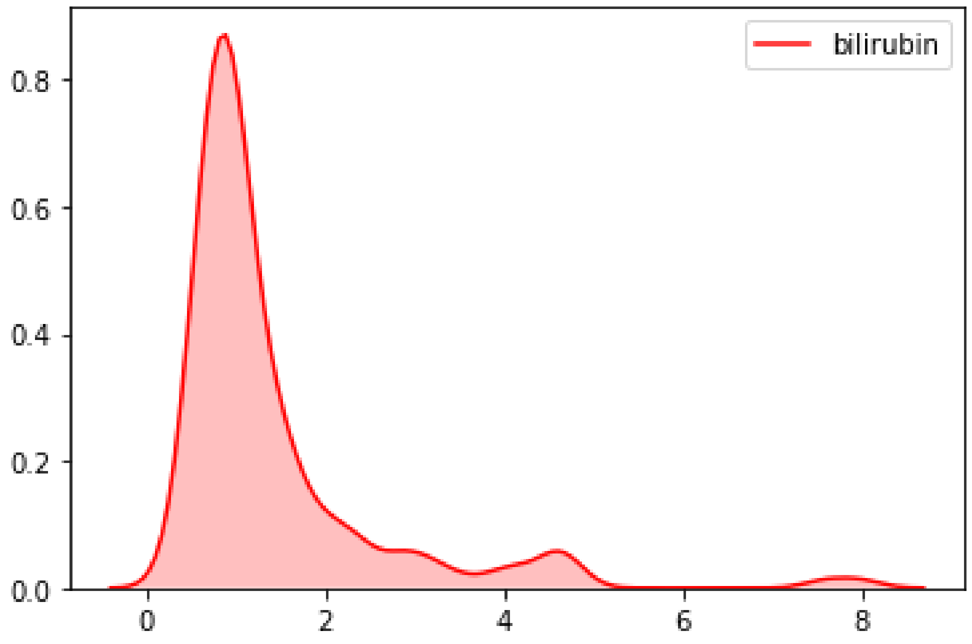
```
[45]: missing_numeric=[var for var in df.columns if df[var].isnull().mean()>0 and
                          df[var].dtypes!='O']
```

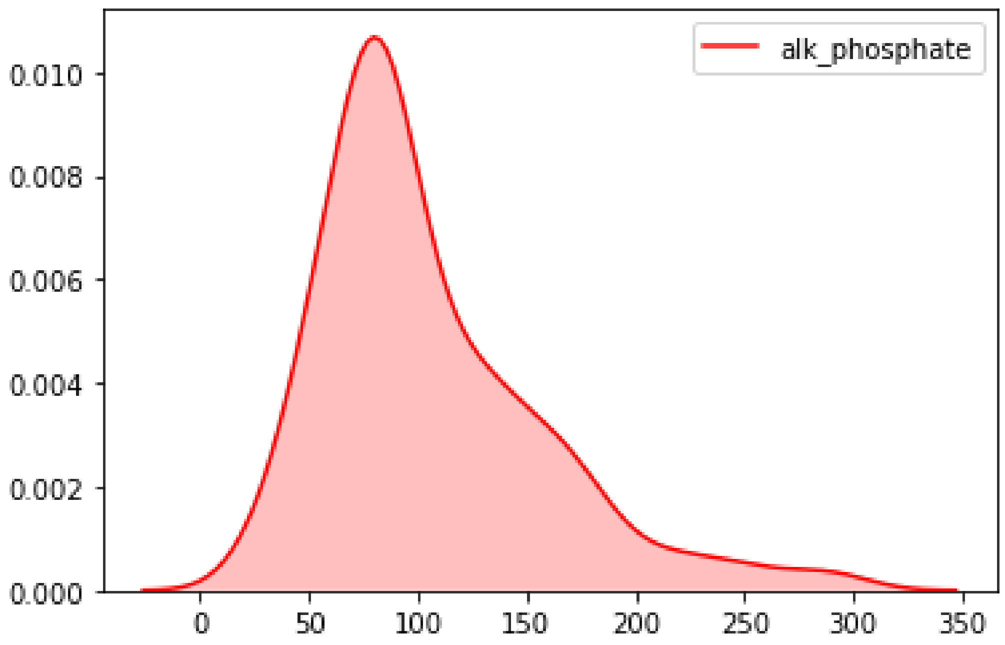
```
[52]: missing_numeric
```

```
[52]: ['bilirubin', 'alk_phosphate', 'sgot', 'albumin', 'protime']
```

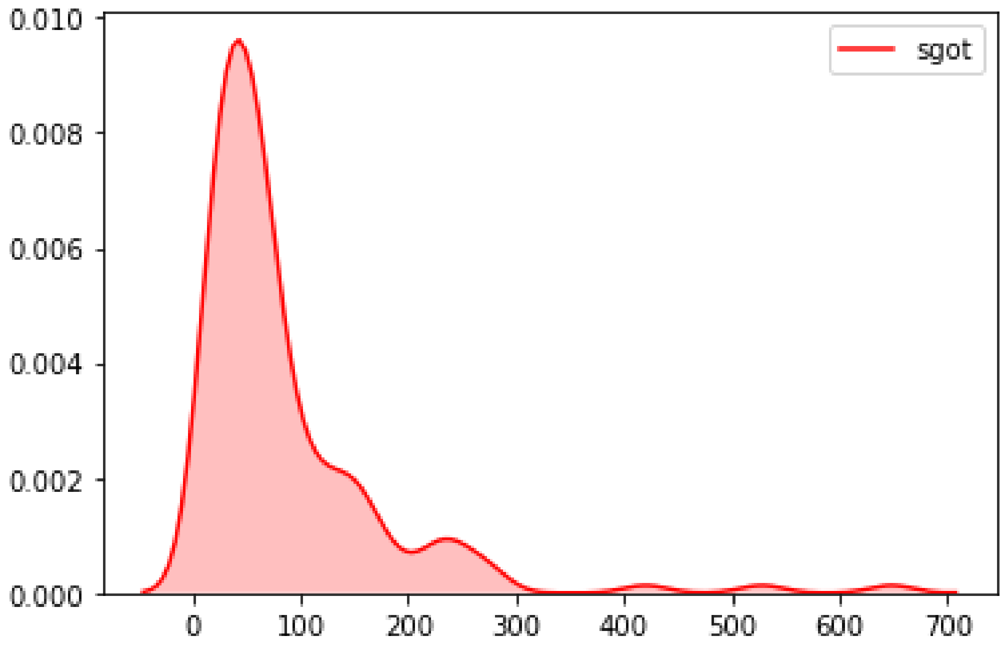
```
[68]: # visualize through the distribution plot
      def distribution_plot(df, var):
          fig=plt.figure()
          ax=fig.add_subplot(111)
          sns.kdeplot(df[var], color='r', shade=True)
          lines,labels=ax.get_legend_handles_labels()
          ax.legend(lines, labels, loc='best')
          plt.show()
```

```
[69]: for var in missing_numeric:
          distribution_plot(df,var)
```

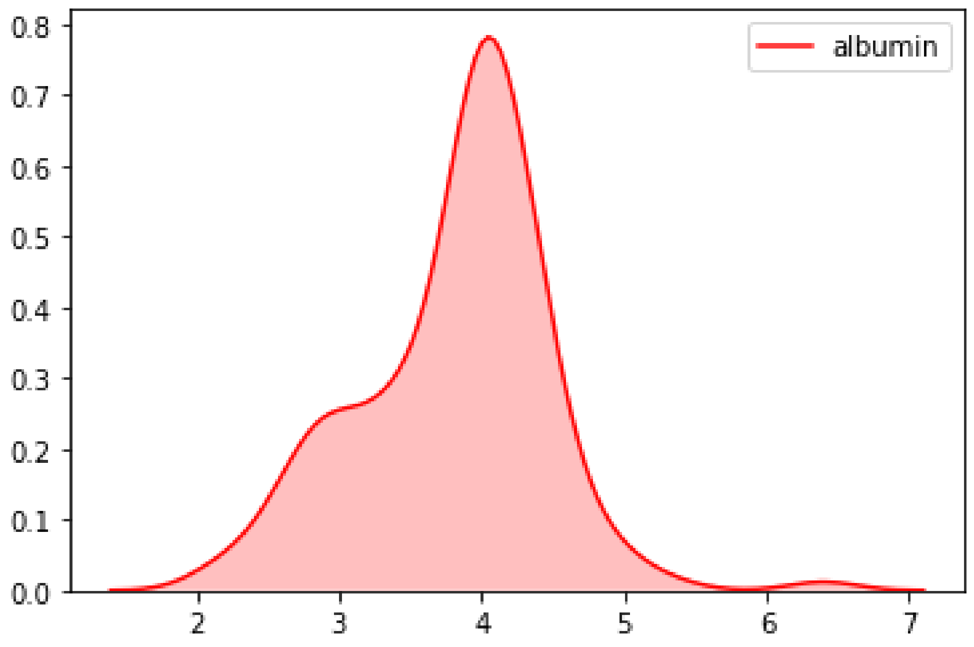


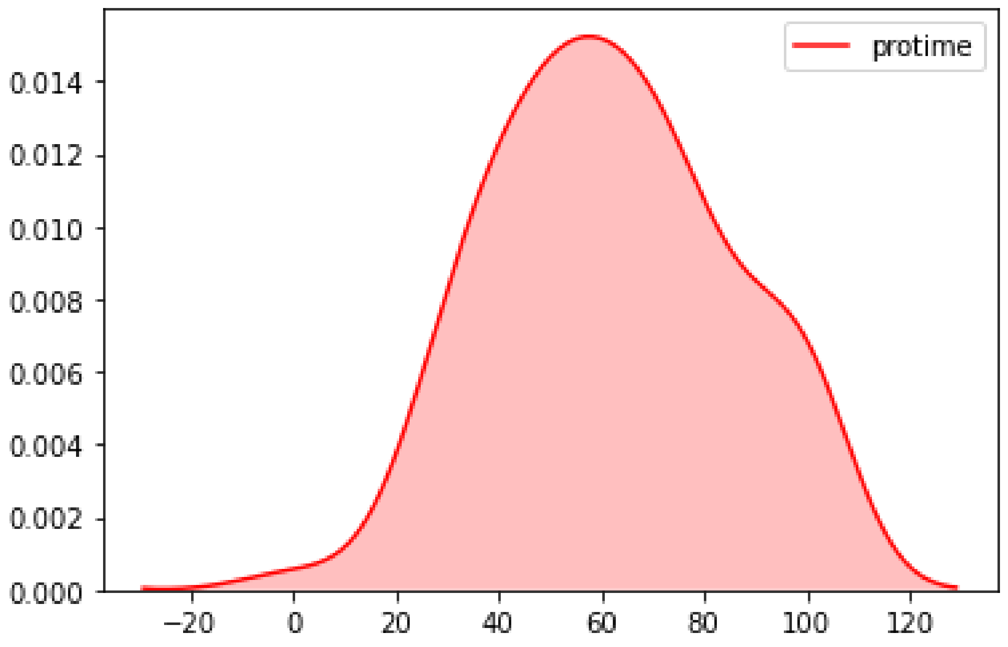


0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250
300
350
alk_phosphate



0.010
0.008
0.006
0.004
0.002
0.000
0
100
200
300
400
500
600
700
sgot





```
[88]: #let's describe a function to impute all the missing values by mean and median␣
       ↪both
      def mean_imputation(df, var):
          df=df.copy()
```

```
    df[var +'_mean_imputed']=df[var].fillna(df[var].mean(), inplace=True)
    return df
```

[89]:
```
df=mean_imputation(df, 'bilirubin')
```

[90]:
```
df['bilirubin'].isnull().sum()
```

[90]:
```
0
```

[ ]:
```
#Likewise we can impute all the numeric columns by mean
#similarly, we can also impute the missing numeric variables by median in case␣
 ↪if our columns have outliers
```

[91]:
```
df=mean_imputation(df, 'alk_phosphate')
df=mean_imputation(df, 'sgot')
df=mean_imputation(df, 'albumin')
df=mean_imputation(df, 'protime')
```