Project By : PRASAD JADHAV

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from imblearn.over_sampling import SMOTE

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import svm

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

In [19]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
dataset = pd.read_csv('chronic_kidney.csv')
dataset.shape
```

Out[2]:
```
(400, 14)
```

In [3]:
```
dataset.head()
```

Out[3]:

	Bp	Sg	Al	Su	Rbc	Bu	Sc	Sod	Pot	Hemo	Wbcc	Rbcc	Htn	Class
0	80.0	1.020	1.0	0.0	1.0	36.0	1.2	137.53	4.63	15.4	7800.0	5.20	1.0	1
1	50.0	1.020	4.0	0.0	1.0	18.0	0.8	137.53	4.63	11.3	6000.0	4.71	0.0	1
2	80.0	1.010	2.0	3.0	1.0	53.0	1.8	137.53	4.63	9.6	7500.0	4.71	0.0	1
3	70.0	1.005	4.0	0.0	1.0	56.0	3.8	111.00	2.50	11.2	6700.0	3.90	1.0	1
4	80.0	1.010	2.0	0.0	1.0	26.0	1.4	137.53	4.63	11.6	7300.0	4.60	0.0	1

In [4]:
```
dataset.tail()
```

Out[4]:

	Bp	Sg	Al	Su	Rbc	Bu	Sc	Sod	Pot	Hemo	Wbcc	Rbcc	Htn	Class
395	80.0	1.020	0.0	0.0	1.0	49.0	0.5	150.0	4.9	15.7	6700.0	4.9	0.0	0
396	70.0	1.025	0.0	0.0	1.0	31.0	1.2	141.0	3.5	16.5	7800.0	6.2	0.0	0
397	80.0	1.020	0.0	0.0	1.0	26.0	0.6	137.0	4.4	15.8	6600.0	5.4	0.0	0
398	60.0	1.025	0.0	0.0	1.0	50.0	1.0	135.0	4.9	14.2	7200.0	5.9	0.0	0
399	80.0	1.025	0.0	0.0	1.0	18.0	1.1	141.0	3.5	15.8	6800.0	6.1	0.0	0

In [5]:
```
print('Number of Rows:',dataset.shape[0])
print('Number of Columns:',dataset.shape[1])
```

```
Number of Rows: 400
Number of Columns: 14
```

In [6]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Bp      400 non-null    float64
 1   Sg      400 non-null    float64
 2   Al      400 non-null    float64
 3   Su      400 non-null    float64
 4   Rbc     400 non-null    float64
 5   Bu      400 non-null    float64
 6   Sc      400 non-null    float64
 7   Sod     400 non-null    float64
 8   Pot     400 non-null    float64
 9   Hemo    400 non-null    float64
 10  Wbcc    400 non-null    float64
 11  Rbcc    400 non-null    float64
 12  Htn     400 non-null    float64
 13  Class   400 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 43.9 KB
```

In [8]:
```
dataset.isna().sum()
```

Out[8]:
```
Bp       0
Sg       0
Al       0
Su       0
Rbc      0
Bu       0
Sc       0
Sod      0
Pot      0
Hemo     0
Wbcc     0
Rbcc     0
Htn      0
Class    0
dtype: int64
```

In [7]:
```
dataset.isnull().sum()
```

Out[7]:
```
Bp       0
Sg       0
Al       0
Su       0
Rbc      0
Bu       0
Sc       0
Sod      0
Pot      0
Hemo     0
Wbcc     0
Rbcc     0
Htn      0
Class    0
dtype: int64
```

In [9]:
```
dataset.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
dataset.describe()
```

Out[10]:

	Bp	Sg	Al	Su	Rbc	Bu	Sc	Sc
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.00000	400.00000	400.0000
mean	76.455000	1.017712	1.015000	0.395000	0.882500	57.40550	3.07235	137.5290
std	13.476536	0.005434	1.272329	1.040038	0.322418	49.28597	5.61749	9.2042
min	50.000000	1.005000	0.000000	0.000000	0.000000	1.50000	0.40000	4.5000
25%	70.000000	1.015000	0.000000	0.000000	1.000000	27.00000	0.90000	135.0000
50%	78.000000	1.020000	1.000000	0.000000	1.000000	44.00000	1.40000	137.5300
75%	80.000000	1.020000	2.000000	0.000000	1.000000	61.75000	3.07000	141.0000
max	180.000000	1.025000	5.000000	5.000000	1.000000	391.00000	76.00000	163.0000

In [12]:
```
dataset.corr()
```

Out[12]:

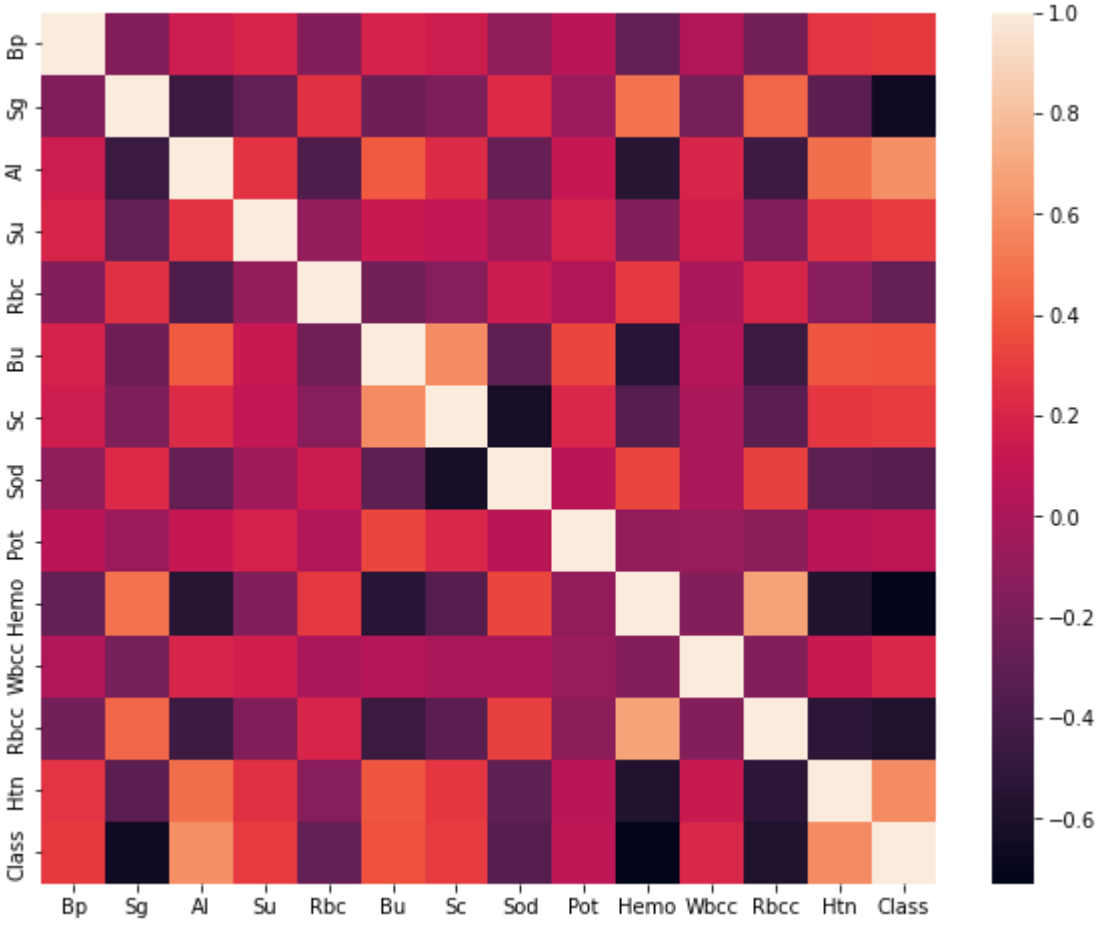
	Bp	Sg	Al	Su	Rbc	Bu	Sc	Sod	P
Bp	1.000000	-0.164057	0.146060	0.190277	-0.151478	0.184173	0.144469	-0.103383	0.0667
Sg	-0.164057	1.000000	-0.460835	-0.292053	0.253894	-0.249263	-0.176141	0.217456	-0.0634
Al	0.146060	-0.460835	1.000000	0.262564	-0.374484	0.405035	0.229396	-0.270709	0.1144
Su	0.190277	-0.292053	0.262564	1.000000	-0.092940	0.126074	0.094568	-0.053448	0.1800
Rbc	-0.151478	0.253894	-0.374484	-0.092940	1.000000	-0.236270	-0.138391	0.140568	0.0181
Bu	0.184173	-0.249263	0.405035	0.126074	-0.236270	1.000000	0.581176	-0.307357	0.3369
Sc	0.144469	-0.176141	0.229396	0.094568	-0.138391	0.581176	1.000000	-0.624493	0.2053
Sod	-0.103383	0.217456	-0.270709	-0.053448	0.140568	-0.307357	-0.624493	1.000000	0.0674
Pot	0.066791	-0.063450	0.114484	0.180098	0.018164	0.336954	0.205361	0.067414	1.0000
Hemo	-0.279441	0.492103	-0.548681	-0.156875	0.280991	-0.540699	-0.342053	0.333604	-0.1006
Wbcc	0.025963	-0.206880	0.200664	0.159033	-0.002205	0.041530	-0.005420	0.006334	-0.0740
Rbcc	-0.220827	0.443437	-0.454131	-0.163825	0.202298	-0.465947	-0.323056	0.316883	-0.1204
Htn	0.268003	-0.318956	0.478309	0.253179	-0.139342	0.387503	0.273904	-0.306501	0.0570
Class	0.290145	-0.659504	0.598389	0.294555	-0.282642	0.371982	0.294076	-0.342268	0.0770

In [11]:
```
dataset.cov()
```

Out[11]:

	Bp	Sg	Al	Su	Rbc	Bu	Sc	S
Bp	181.617018	-0.012014	2.504436	2.666942	-0.658183	122.328318	10.936923	-12.8238
Sg	-0.012014	0.000030	-0.003186	-0.001651	0.000445	-0.066758	-0.005377	0.0108
Al	2.504436	-0.003186	1.618822	0.347444	-0.153622	25.398915	1.639564	-3.1702
Su	2.666942	-0.001651	0.347444	1.081679	-0.031165	6.462484	0.552503	-0.5116
Rbc	-0.658183	0.000445	-0.153622	-0.031165	0.103954	-3.754490	-0.250651	0.4171
Bu	122.328318	-0.066758	25.398915	6.462484	-3.754490	2429.106887	160.906503	-139.4300
Sc	10.936923	-0.005377	1.639564	0.552503	-0.250651	160.906503	31.556195	-32.2893
Sod	-12.823866	0.010876	-3.170236	-0.511644	0.417153	-139.430037	-32.289372	84.7186
Pot	2.538124	-0.000972	0.410734	0.528170	0.016514	46.828385	3.252943	1.7496
Hemo	-10.228812	0.007263	-1.896169	-0.443158	0.246076	-72.382890	-5.219070	8.3402
Wbcc	882.846165	-2.836586	644.204160	417.342807	-1.793910	5164.622561	-76.821766	147.1115
Rbcc	-2.500767	0.002025	-0.485538	-0.143177	0.054809	-19.297544	-1.524972	2.4509
Htn	1.740948	-0.000835	0.293343	0.126924	-0.021656	9.205893	0.741667	-1.3598
Class	1.895363	-0.001737	0.369048	0.148496	-0.044173	8.886779	0.800758	-1.5270

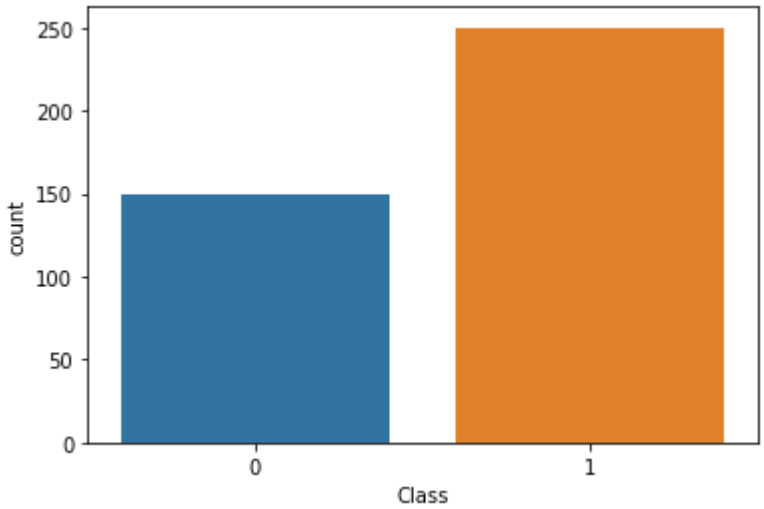
In [16]:
```
plt.figure(figsize=(10,8))
sns.heatmap(dataset.corr())
plt.show()
```



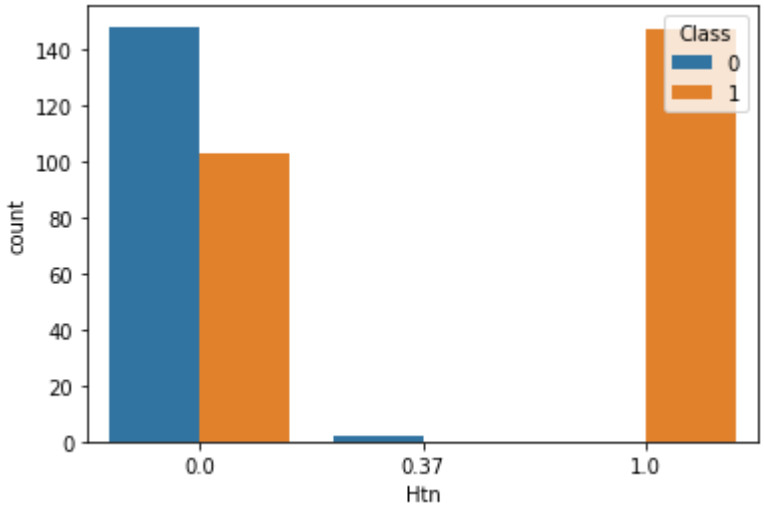
In [17]:
```
dataset['Class'].value_counts()
```

Out[17]:
```
1    250
0    150
Name: Class, dtype: int64
```

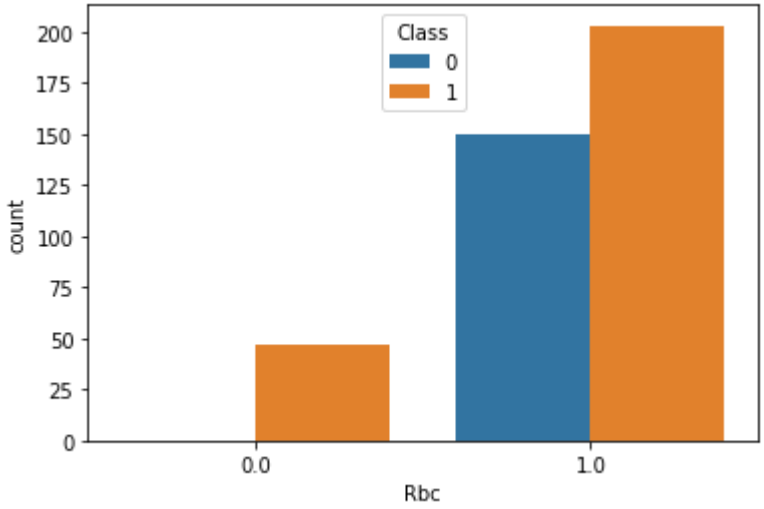
In [20]:
```
sns.countplot(dataset['Class'])
plt.show()
```



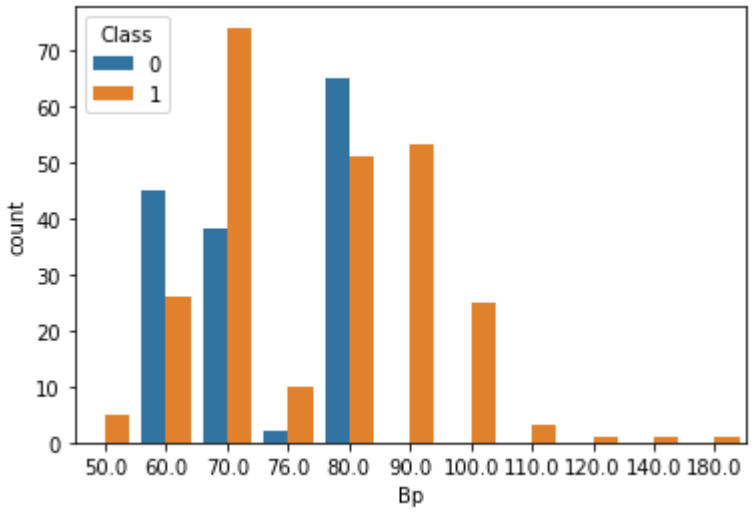
In [24]:
```
sns.countplot(dataset['Htn'],hue=dataset['Class'])
plt.show()
```



In [25]:
```
sns.countplot(dataset['Rbc'],hue=dataset['Class'])
plt.show()
```



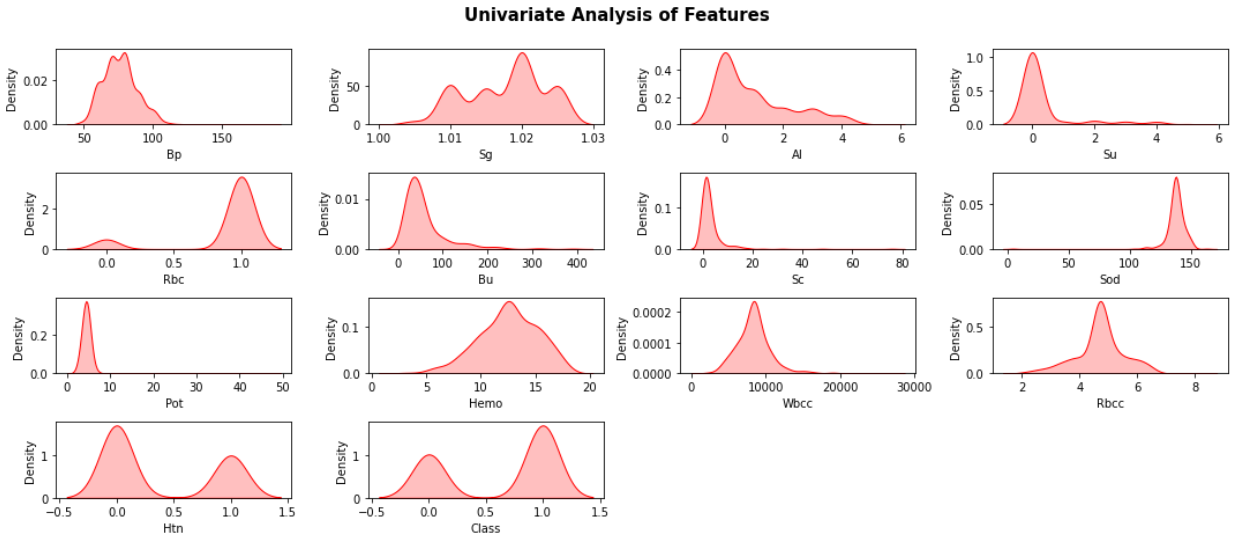
In [28]:
```
sns.countplot(dataset['Bp'],hue=dataset['Class'])
plt.show()
```



In [33]:
```
num_features = [feature for feature in dataset.columns]
```

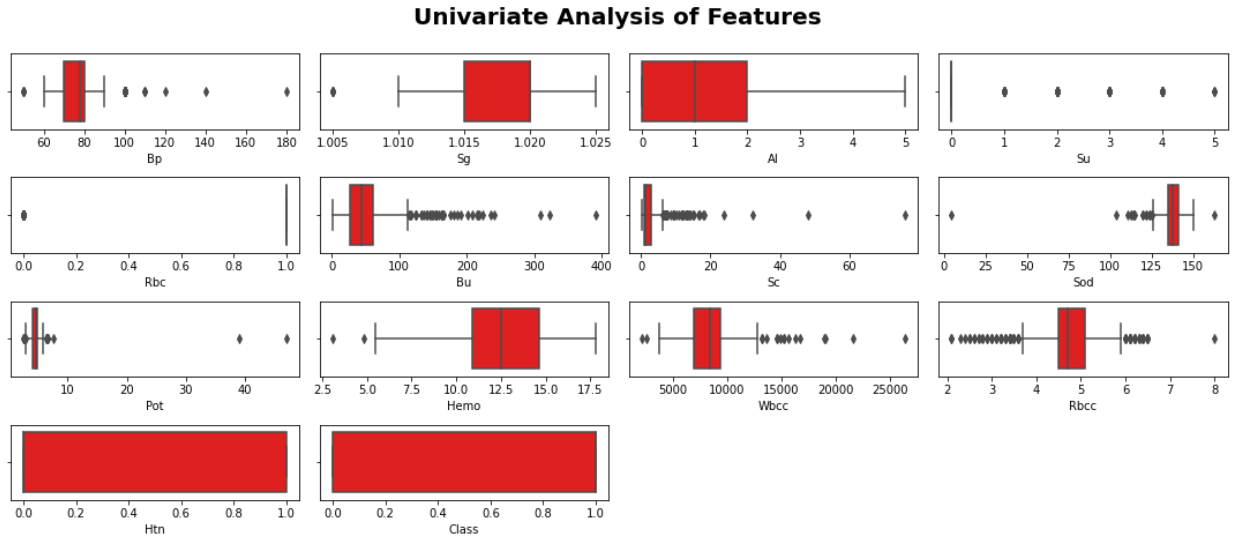
In [34]:
```
plt.figure(figsize=(15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=15,y=

for i in range(0,len(num_features)):
    plt.subplot(10,4,i+1)
    sns.kdeplot(x=dataset[num_features[i]],shade=True,color='red')
    plt.tight_layout()
```



In [35]:
```
plt.figure(figsize = (15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=20,y=

for i in range(0,len(num_features)):
    plt.subplot(10,4,i+1)
    sns.boxplot(data=dataset,x=features[i],color='red')
    plt.xlabel(num_features[i])
    plt.tight_layout()
```



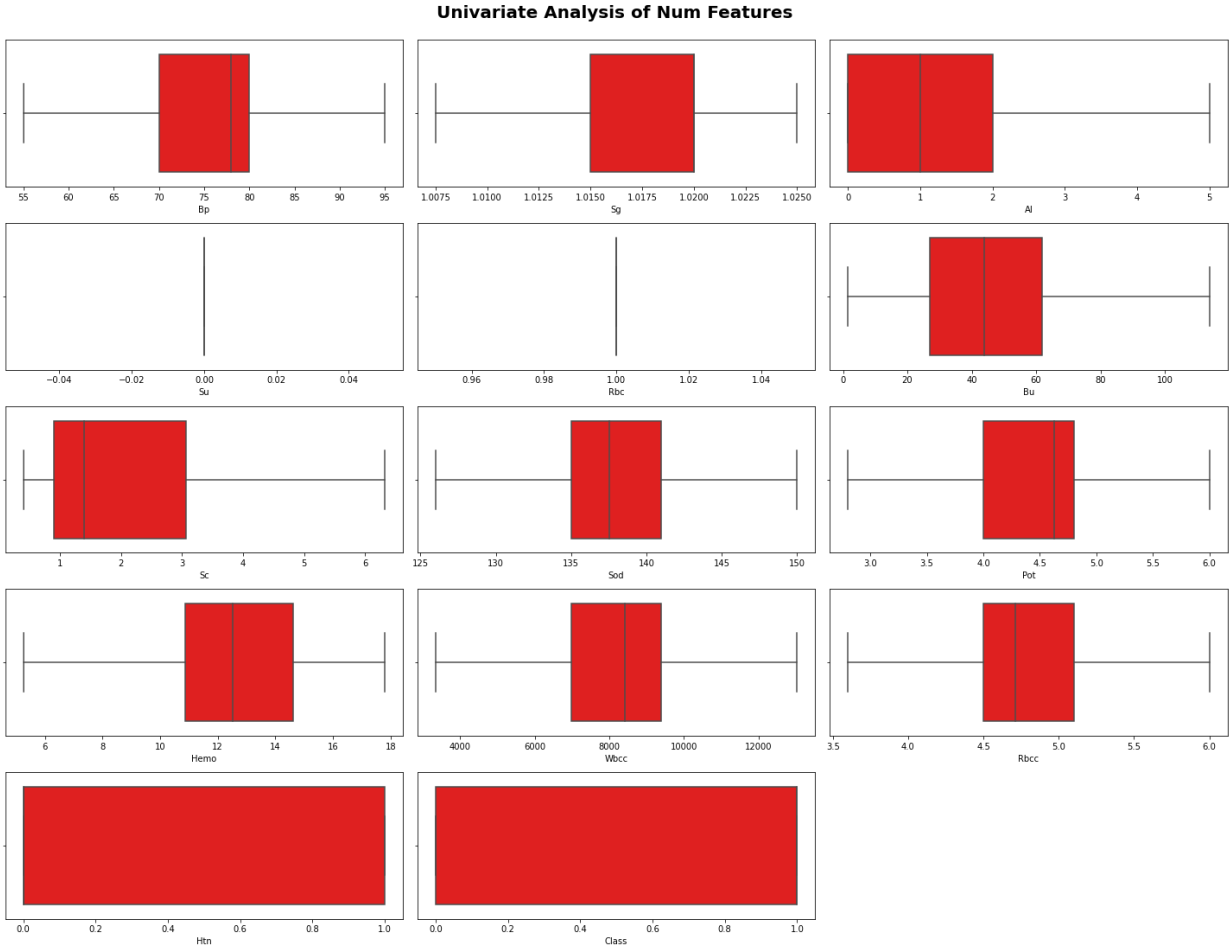
In [36]:
```
def remove_outliers(in_dataset, in_cols):

    first_quartile = in_dataset[in_cols].quantile(0.25)
    third_quartile = in_dataset[in_cols].quantile(0.75)
    iqr = third_quartile - first_quartile
    upper_limit = third_quartile + 1.5 * iqr
    lower_limit = first_quartile - 1.5 * iqr
```

```
    in_dataset.loc[(in_dataset[in_cols] > upper_limit), in_cols] = upper_limit
    in_dataset.loc[(in_dataset[in_cols] < lower_limit), in_cols] = lower_limit
    return in_dataset
```

In [37]:
```
for features in num_features:
    dataset = remove_outliers(dataset,features)
```

In [39]:
```
plt.figure(figsize = (20,250))
plt.suptitle('Univariate Analysis of Num Features',fontweight='bold',fontsize=
for i in range(0,len(num_features)):
    plt.subplot(85,3,i+1)
    sns.boxplot(data=dataset,x=num_features[i],color='red')
    plt.xlabel(num_features[i])
    plt.tight_layout()
```



In [40]:
```
dataset['Class'].value_counts()
```

Out[40]:
```
1.0    250
0.0    150
Name: Class, dtype: int64
```

In [43]:
```
dataset['Class'] = dataset['Class'].astype('int64')
```

In [44]:
```
dataset['Class'].value_counts()
```

Out[44]:
```
1    250
0    150
Name: Class, dtype: int64
```

In [46]:
```
X = dataset.drop('Class',axis=1)
y = dataset['Class']
```

In [47]:
```
X_res,y_res = SMOTE().fit_resample(X,y)
```

In [48]:
```
X_train,X_test,y_train,y_test = train_test_split(X_res,y_res,test_size=0.20,ra
```

In [49]:
```
st = StandardScaler()
X_train = st.fit_transform(X_train)
X_test = st.fit_transform(X_test)
```

In [50]:
```
model_df = {}

def model_val(model,X,y):
    X_train,X_test,y_train,y_test = train_test_split(X_res,y_res,test_size=0.2(
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print(f'{model} Accuracy is {accuracy_score(y_test,y_pred)}')

    score = cross_val_score(model,X,y,cv=5,n_jobs=-1)
    print(f'{model} Average cross val score is {np.mean(score)}')
    model_df[model] = round(np.mean(score)*100,2)
```

In [51]:
```
model = LogisticRegression()
model_val(model,X,y)
```

```
LogisticRegression() Accuracy is 0.94
LogisticRegression() Average cross val score is 0.9549999999999998
```

In [52]:
```
model = DecisionTreeClassifier()
model_val(model,X,y)
```

```
DecisionTreeClassifier() Accuracy is 0.98
DecisionTreeClassifier() Average cross val score is 0.97
```

In [53]:
```
model = RandomForestClassifier()
model_val(model,X,y)
```

```
RandomForestClassifier() Accuracy is 1.0
RandomForestClassifier() Average cross val score is 0.9824999999999999
```

In [54]:
```
model = GradientBoostingClassifier()
model_val(model,X,y)
```

```
GradientBoostingClassifier() Accuracy is 0.99
GradientBoostingClassifier() Average cross val score is 0.9800000000000001
```

In [55]:
```
model_df
```

Out[55]:
```
{LogisticRegression(): 95.5,
 DecisionTreeClassifier(): 97.0,
 RandomForestClassifier(): 98.25,
 GradientBoostingClassifier(): 98.0}
```

In [67]:
```
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
```

Out[67]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [68]:
```
y_pred = rf.predict(X_test)
```

In [69]:
```
accuracy_score(y_test,y_pred)
```

Out[69]:
```
0.99
```

In [56]:
```
import pickle
import joblib
```

In [70]:
```
pickle.dump(rf,open('chronic_kidney.pkl','wb'))
```

In [71]:
```
model = pickle.load(open('chronic_kidney.pkl','rb'))
```

In [72]:
```
new_df = pd.DataFrame({
    'Bp':80.0,
    'Sg':1.0200,
    'Al':1.0,
    'Su':0.0,
    'Rbc':1.0,
    'Bu':36.0,
    'Sc':1.2,
    'Sod':137.53,
    'Pot':4.63,
    'Hemo':15.4,
    'Wbcc':7800.0,
    'Rbcc':5.20,
    'Htn':1.0
},index=[0])
```

In [73]:
```
model.predict(new_df)
```

Out[73]:
```
array([1], dtype=int64)
```

In [74]:
```
p = model.predict(new_df)

prob = model.predict_proba(new_df)
if p == 1:
    print('Chronic Kidney Disease!')
    print(f'You will be Chronic Kidney Disease! with Probability of {prob[0][1]
else:
    print('Not-Chronic Kidney Disease!')
```

```
Chronic Kidney Disease!
You will be Chronic Kidney Disease! with Probability of 0.82
```

CONNECT WITH ME :

LinkedIn GitHub kaggle Medium

PRASADMJADHAV2