```
import pandas as pd
import io
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.impute import KNNImputer
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler


from google.colab import files


uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving chronic_kidney_disease.csv to chronic_kidney_disease (1).csv

```
df = pd.read_csv(io.BytesIO(uploaded['chronic_kidney_disease.csv']))


X=df.iloc[:,0:24]
Y=df["class"]


df.head()
```

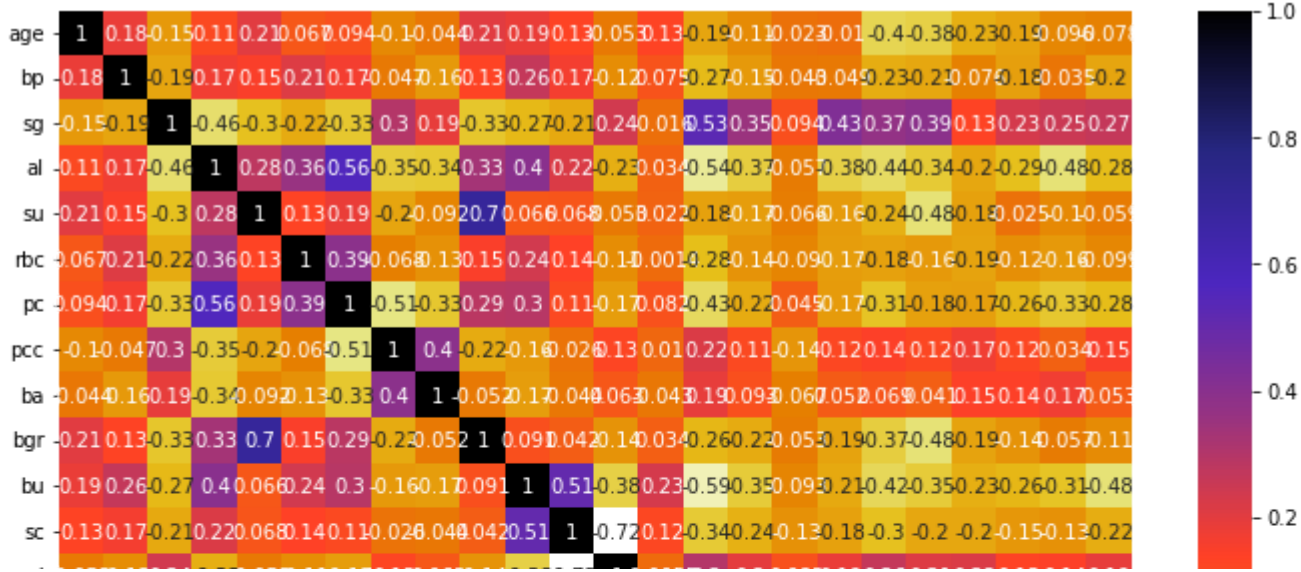
| | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | ... | pcv | wbcc | rbcc | htn | dm | cad | appet | pe | ane | class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 48 | 80 | 1.02 | 1 | 0 | 0 | 0 | 1 | 1 | 121 | | 44 | 7800 | 5.2 | 0 | 0 | 1 | 1 | 1 | 1 | ckd |

```
knn_missing_values_imputer = KNNImputer(n_neighbors=5)
df = df.replace('?', np.nan)
X = pd.DataFrame(knn_missing_values_imputer.fit_transform(X),columns = X.columns)
standard_feature_scaler = StandardScaler()
X = standard_feature_scaler.fit_transform(X)
X = pd.DataFrame(X, columns=['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu', 'sc', 'sod', 'pot',
                                                        'hemo', 'pcv', 'wbcc', 'rbcc', 'htn', 'dm',
                                                        'cad', 'appet', 'pe', 'ane'])

X_train, X_test, Y_train,Y_test = train_test_split(X, Y, train_size = 0.7, test_size = 0.3)



import seaborn as sns
plt.figure(figsize=(12,10))
cor=X_train.corr()
sns.heatmap(cor,annot=True,cmap=plt.cm.CMRmap_r)
plt.show()
```




```
def correlate(dataset,threshold):
  col_corr=set()
  corr_matrix=dataset.corr()
  for i in range(len(corr_matrix.columns)):
    for j in range(i):
      if abs(corr_matrix.iloc[i,j])>threshold:
        colname=corr_matrix.columns[i]
        col_corr.add(colname)
  return col_corr
```

```
corr_features=correlate(X_train,0.6)
print(len(set(corr_features)))
print(corr_features)
```

```
4
{'rbcc', 'pcv', 'sod', 'bgr'}
```

```
X_train.drop(corr_features,axis=1)
X_test.drop(corr_features,axis=1)
```

| | age | bp | sg | al | su | rbc | pc | pcc | ba | bu | sc | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **291** | -0.257518 | 0.264037 | 1.413392 | -0.795017 | -0.431146 | -0.364890 | -0.484322 | 0.360477 | 0.263664 | -0.268332 | -0.367591 | 0. |
| **250** | -0.668232 | 0.264037 | 1.413392 | -0.795017 | -0.431146 | -0.364890 | -0.484322 | 0.360477 | 0.263664 | -0.955146 | -0.332030 | 0. |
| **107** | 0.211870 | 1.745312 | -0.409164 | -0.023156 | 3.388529 | -0.364890 | -0.484322 | 0.360477 | 0.263664 | -0.086528 | -0.047541 | -0. |
| **55** | -0.961600 | 0.264037 | -2.231720 | 1.520567 | -0.431146 | 2.740554 | -0.484322 | 0.360477 | 0.263664 | 0.123556 | -0.043985 | -0. |
| **38** | 1.033299 | 0.264037 | 0.502114 | 1.520567 | -0.431146 | 2.740554 | -0.484322 | 0.360477 | 0.263664 | 0.923492 | 0.183606 | 0. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **54** | 0.681258 | 0.264037 | -1.320442 | 0.748706 | 1.478692 | -0.364890 | -0.484322 | 0.360477 | 0.263664 | -0.240052 | 0.059143 | -0. |
| **352** | -0.844253 | -1.217238 | 0.502114 | -0.795017 | -0.431146 | -0.364890 | -0.484322 | 0.360477 | 0.263664 | -0.207731 | -0.349810 | 0. |
| **325** | 0.387890 | 0.264037 | 0.502114 | -0.795017 | -0.431146 | -0.364890 | -0.484322 | 0.360477 | 0.263664 | -0.147130 | -0.332030 | -0. |
| **12** | 0.974625 | -0.476600 | -0.409164 | 1.520567 | 0.523773 | -0.364890 | -0.484322 | -2.774104 | 0.263664 | 0.297279 | -0.172005 | 0. |
| **0** | -0.198844 | 0.264037 | 0.502114 | -0.023156 | -0.431146 | -0.364890 | -0.484322 | 0.360477 | 0.263664 | -0.429935 | -0.332030 | -0. |

120 rows × 20 columns

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report


linear_svm=SVC(C=1.0,kernel='linear',random_state=0)
linear_svm=linear_svm.fit(X_train,Y_train)


print(linear_svm.score(X_train,Y_train))
print(linear_svm.score(X_test,Y_test))
```

```
1.0
0.975
```

```
predicted =linear_svm.predict(X_test)
report = classification_report(Y_test, predicted)
```

```
print(report)
```

```
                  precision    recall  f1-score   support

             ckd       1.00      0.96      0.98        81
          notckd       0.93      1.00      0.96        39

        accuracy                           0.97       120
       macro avg       0.96      0.98      0.97       120
    weighted avg       0.98      0.97      0.98       120
```