```
# Importing Libraries:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


dataset=pd.read_csv(r"/content/kidney_disease.csv")



# Top 5 records:
dataset.head()
```

| | id | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | bu | sc | sod | pot | hemo | pcv | wc | rc | htn |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | NaN | normal | notpresent | notpresent | 121.0 | 36.0 | 1.2 | NaN | NaN | 15.4 | 44 | 7800 | 5.2 | yes |
| **1** | 1 | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | NaN | normal | notpresent | notpresent | NaN | 18.0 | 0.8 | NaN | NaN | 11.3 | 38 | 6000 | NaN | no |
| **2** | 2 | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | normal | normal | notpresent | notpresent | 423.0 | 53.0 | 1.8 | NaN | NaN | 9.6 | 31 | 7500 | NaN | no |
| **3** | 3 | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | normal | abnormal | present | notpresent | 117.0 | 56.0 | 3.8 | 111.0 | 2.5 | 11.2 | 32 | 6700 | 3.9 | yes |
| **4** | 4 | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | normal | normal | notpresent | notpresent | 106.0 | 26.0 | 1.4 | NaN | NaN | 11.6 | 35 | 7300 | 4.6 | no |



```
# Dropping unneccsary feature :
dataset = dataset.drop('id', axis=1)


# Shape of dataset:
dataset.shape
```

```
(400, 25)
```

```
# Cheaking Missing (NaN) Values:
dataset.isnull().sum()
```

```
age               9
bp               12
sg               47
al               46
su               49
rbc             152
pc               65
pcc               4
ba                4
bgr              44
bu               19
sc               17
sod              87
pot              88
hemo             52
pcv              70
wc              105
rc              130
htn               2
dm                2
cad               2
appet             1
pe                1
ane               1
classification    0
```



```
# Description:
dataset.describe()
```

| | age | bp | sg | al | su | bgr | bu | sc | sod | pot | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 391.000000 | 388.000000 | 353.000000 | 354.000000 | 351.000000 | 356.000000 | 381.000000 | 383.000000 | 313.000000 | 312.000000 | 348.0( |
| **mean** | 51.483376 | 76.469072 | 1.017408 | 1.016949 | 0.450142 | 148.036517 | 57.425722 | 3.072454 | 137.528754 | 4.627244 | 12.5 |
| **std** | 17.169714 | 13.683637 | 0.005717 | 1.352679 | 1.099191 | 79.281714 | 50.503006 | 5.741126 | 10.408752 | 3.193904 | 2.9 |
| **min** | 2.000000 | 50.000000 | 1.005000 | 0.000000 | 0.000000 | 22.000000 | 1.500000 | 0.400000 | 4.500000 | 2.500000 | 3.1( |
| **25%** | 42.000000 | 70.000000 | 1.010000 | 0.000000 | 0.000000 | 99.000000 | 27.000000 | 0.900000 | 135.000000 | 3.800000 | 10.3( |

```
# Datatypes:
dataset.dtypes
```

```
age               float64
bp                float64
sg                float64
al                float64
su                float64
rbc                object
pc                 object
pcc                object
ba                 object
bgr               float64
bu                float64
sc                float64
sod               float64
```

Saved successfully!

```
wc                 object
rc                 object
htn                object
dm                 object
cad                object
appet              object
pe                 object
ane                object
classification     object
dtype: object
```

```
dataset.head()
```

| | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | ... | pcv | wc | rc | htn | dm | cad | appet | pe | ane |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | NaN | normal | notpresent | notpresent | 121.0 | ... | 44 | 7800 | 5.2 | yes | yes | no | good | no | no |
| **1** | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | NaN | normal | notpresent | notpresent | NaN | ... | 38 | 6000 | NaN | no | no | no | good | no | no |
| **2** | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | normal | normal | notpresent | notpresent | 423.0 | ... | 31 | 7500 | NaN | no | yes | no | poor | no | yes |
| **3** | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | normal | abnormal | present | notpresent | 117.0 | ... | 32 | 6700 | 3.9 | yes | no | no | poor | yes | yes |
| **4** | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | normal | normal | notpresent | notpresent | 106.0 | ... | 35 | 7300 | 4.6 | no | no | no | good | no | no |

5 rows × 25 columns

```
dataset['rbc'].value_counts()
```

```
normal      201
abnormal     47
Name: rbc, dtype: int64
```

```
dataset['rbc'] = dataset['rbc'].replace(to_replace = {'normal' : 0, 'abnormal' : 1})
```

Saved successfully!

```
abnormal     76
Name: pc, dtype: int64
```

```
dataset['pc'] = dataset['pc'].replace(to_replace = {'normal' : 0, 'abnormal' : 1})
```

```
dataset['pcc'].value_counts()
```

```
notpresent    354
```

```
present        42
Name: pcc, dtype: int64
```

```
dataset['pcc'] = dataset['pcc'].replace(to_replace = {'notpresent':0,'present':1})
```

```
dataset['ba'].value_counts()
```

```
notpresent    374
present        22
Name: ba, dtype: int64
```

```
dataset['ba'] = dataset['ba'].replace(to_replace = {'notpresent':0,'present':1})
```

```
dataset['htn'].value_counts()
```

```
no     251
yes    147
Name: htn, dtype: int64
```

```
dataset['htn'] = dataset['htn'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

```
dataset['dm'].value_counts()
```

Saved successfully!

```
\tno       3
\tyes      2
 yes       1
Name: dm, dtype: int64
```

```
dataset['dm'] = dataset['dm'].replace(to_replace = {'\tyes':'yes', ' yes':'yes', '\tno':'no'})
```

```
dataset['dm'] = dataset['dm'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

```
dataset['cad'].value_counts()
```

```
no      362
yes      34
\tno      2
Name: cad, dtype: int64
```

```
dataset['cad'] = dataset['cad'].replace(to_replace = {'\tno':'no'})
```

```
dataset['cad'] = dataset['cad'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

```
dataset['appet'].unique()
```

```
array(['good', 'poor', nan], dtype=object)
```

```
dataset['appet'] = dataset['appet'].replace(to_replace={'good':1,'poor':0,'no':np.nan})
```

```
dataset['pe'].value_counts()
```

```
no     323
yes     76
```

Saved successfully!

```
dataset['pe'] = dataset['pe'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

```
dataset['ane'].value_counts()
```

```
no     339
yes     60
Name: ane, dtype: int64
```

```
dataset['ane'] = dataset['ane'].replace(to_replace = {'yes' : 1, 'no' : 0})

dataset['classification'].value_counts()
```

```
ckd       248
notckd    150
ckd\t       2
Name: classification, dtype: int64
```

```
dataset['classification'] = dataset['classification'].replace(to_replace={'ckd\t':'ckd'})


dataset["classification"] = [1 if i == "ckd" else 0 for i in dataset["classification"]]


dataset.head()
```

| | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | ... | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | NaN | 0.0 | 0.0 | 0.0 | 121.0 | ... | 44 | 7800 | 5.2 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1 |
| **1** | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | NaN | 0.0 | 0.0 | 0.0 | NaN | ... | 38 | 6000 | NaN | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1 |
| **2** | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 423.0 | ... | 31 | 7500 | NaN | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1 |
| **3** | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 117.0 | ... | 32 | 6700 | 3.9 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1 |
| **4** | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 106.0 | ... | 35 | 7300 | 4.6 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1 |

Saved successfully!

```
# Datatypes:
dataset.dtypes
```

```
age               float64
bp                float64
sg                float64
al                float64
su                float64
```

```
rbc               float64
pc                float64
pcc               float64
ba                float64
bgr               float64
bu                float64
sc                float64
sod               float64
pot               float64
hemo              float64
pcv                object
wc                 object
rc                 object
htn               float64
dm                float64
cad               float64
appet             float64
pe                float64
ane               float64
classification      int64
dtype: object
```

```
dataset['pcv'] = pd.to_numeric(dataset['pcv'], errors='coerce')
dataset['wc'] = pd.to_numeric(dataset['wc'], errors='coerce')
dataset['rc'] = pd.to_numeric(dataset['rc'], errors='coerce')
```

Saved successfully!

```
age               float64
bp                float64
sg                float64
al                float64
su                float64
rbc               float64
pc                float64
pcc               float64
ba                float64
bgr               float64
```

```
bu                float64
sc                float64
sod               float64
pot               float64
hemo              float64
pcv               float64
wc                float64
rc                float64
htn               float64
dm                float64
cad               float64
appet             float64
pe                float64
ane               float64
classification      int64
dtype: object
```

```
# Description:
dataset.describe()
```

| | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|

```
# Cheaking Missing (NaN) Values:
dataset.isnull().sum().sort_values(ascending=False)
```

```
rbc               152
rc                131
wc                106
pot                88
sod                87
pcv                71
pc                 65
hemo               52
su                 49
sg                 47
al                 46
bgr                44
bu                 19
sc                 17
bp                 12
age                 9
ba                  4
pcc                 4
htn                 2
dm                  2
cad                 2
classification      0
dtype: int64
```



```
dataset.columns
```

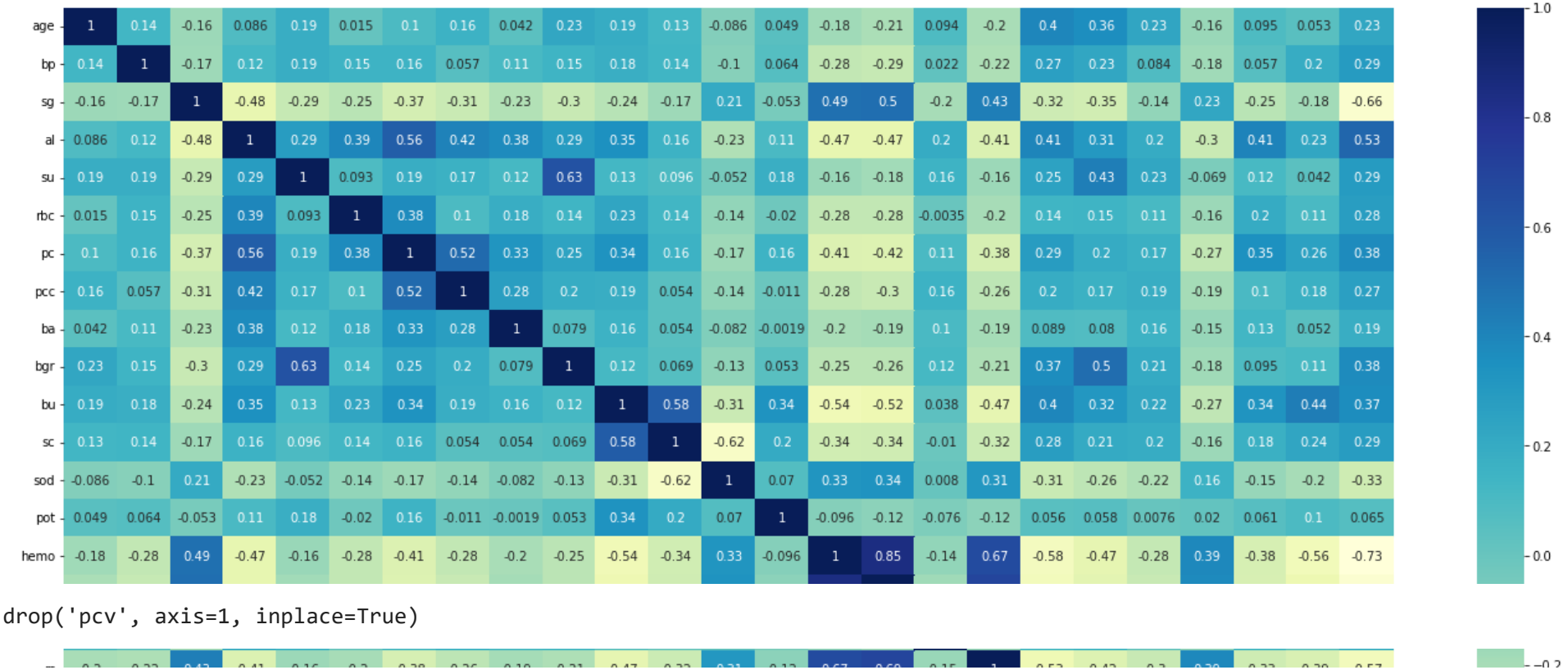
```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
       'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

```
features = ['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
           'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
           'appet', 'pe', 'ane']


for feature in features:
    dataset[feature] = dataset[feature].fillna(dataset[feature].median())


dataset.isnull().any().sum()
```

```
0
```

```
plt.figure(figsize=(24,14))
sns.heatmap(dataset.corr(), annot=True, cmap='YlGnBu')
plt.show()
```

Saved successfully!




```
dataset.drop('pcv', axis=1, inplace=True)
```

```
dataset.head()
```

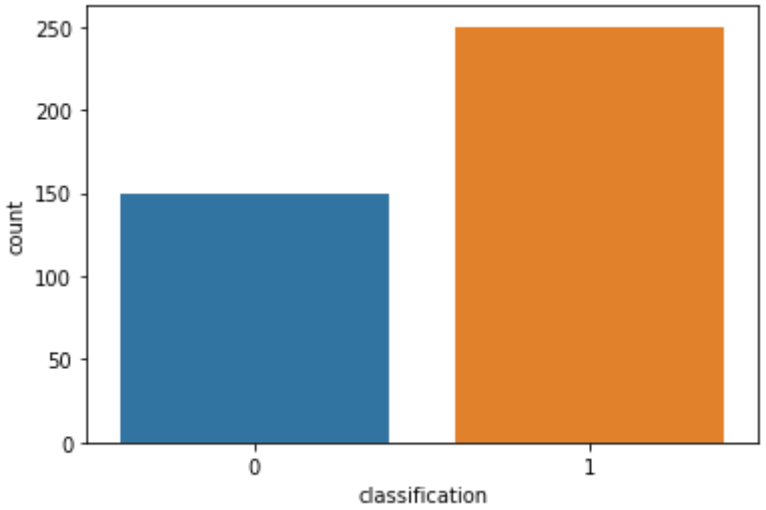
| | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | ... | hemo | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 0.0 | 0.0 | 0.0 | 0.0 | 121.0 | ... | 15.4 | 7800.0 | 5.2 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1 |
| **1** | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 121.0 | ... | 11.3 | 6000.0 | 4.8 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1 |
| **2** | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 423.0 | ... | 9.6 | 7500.0 | 4.8 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1 |
| **3** | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 117.0 | ... | 11.2 | 6700.0 | 3.9 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1 |
| **4** | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 106.0 | ... | 11.6 | 7300.0 | 4.6 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1 |

5 rows × 24 columns

Saved successfully!

```
# Target feature:
sns.countplot(dataset['classification'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82460d0790>
```

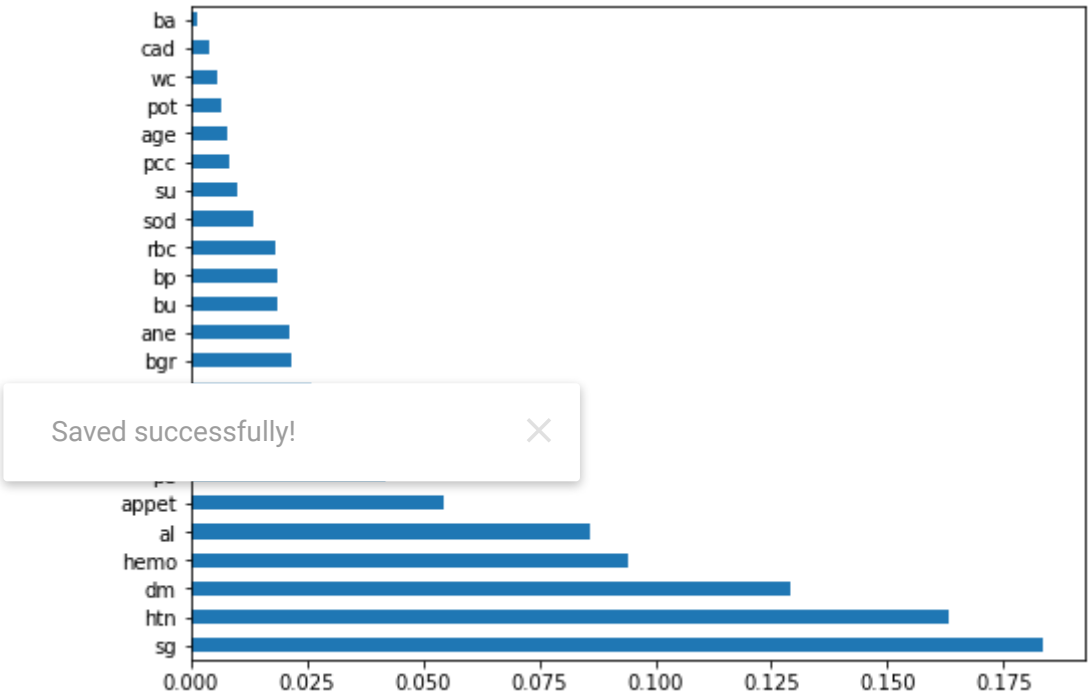



```
# Independent and Dependent Feature:
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
```

Saved successfully!

| | age | bp | sg | al | su | rbc | pc | pcc | ba | bgr | ... | pot | hemo | wc | rc | htn | dm | cad | appet | pe | ane |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 121.0 | ... | 4.4 | 15.4 | 7800.0 | 5.2 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 |

```
# Feature Importance:
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model=ExtraTreesClassifier()
model.fit(X,y)

plt.figure(figsize=(8,6))
ranked_features=pd.Series(model.feature_importances_,index=X.columns)
ranked_features.nlargest(24).plot(kind='barh')
plt.show()
```




Saved successfully!

```
ranked_features.nlargest(8).index
```

```
Index(['sg', 'htn', 'dm', 'hemo', 'al', 'appet', 'pe', 'pc'], dtype='object')
```

```
X = dataset[['sg', 'htn', 'hemo', 'dm', 'al', 'appet', 'rc', 'pc']]
X.head()
```

| | sg | htn | hemo | dm | al | appet | rc | pc |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.020 | 1.0 | 15.4 | 1.0 | 1.0 | 1.0 | 5.2 | 0.0 |
| **1** | 1.020 | 0.0 | 11.3 | 0.0 | 4.0 | 1.0 | 4.8 | 0.0 |
| **2** | 1.010 | 0.0 | 9.6 | 1.0 | 2.0 | 0.0 | 4.8 | 0.0 |
| **3** | 1.005 | 1.0 | 11.2 | 0.0 | 4.0 | 0.0 | 3.9 | 1.0 |
| **4** | 1.010 | 0.0 | 11.6 | 0.0 | 2.0 | 1.0 | 4.6 | 0.0 |

```
X.tail()
```

| | sg | htn | hemo | dm | al | appet | rc | pc |
|---|---|---|---|---|---|---|---|---|
| **395** | 1.020 | 0.0 | 15.7 | 0.0 | 0.0 | 1.0 | 4.9 | 0.0 |
| **396** | 1.025 | 0.0 | 16.5 | 0.0 | 0.0 | 1.0 | 6.2 | 0.0 |
| **397** | 1.020 | 0.0 | 15.8 | 0.0 | 0.0 | 1.0 | 5.4 | 0.0 |
| | | | | | | 1.0 | 5.9 | 0.0 |
| **399** | 1.025 | 0.0 | 15.8 | 0.0 | 0.0 | 1.0 | 6.1 | 0.0 |

Saved successfully!

```
y.head()
```

```
0    1
1    1
2    1
3    1
```

```
4    1
Name: classification, dtype: int64
```

```
# Train Test Split:
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, random_state=33)


print(X_train.shape)
print(X_test.shape)
```

```
(280, 8)
(120, 8)
```

```
# Importing Performance Metrics:
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


# RandomForestClassifier:
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(X_train,y_train)

# Predictions:
y_pred = RandomForest.predict(X_test)
print(y_pred)
```



```
test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0
 1 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0
 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 0
 1 1 1 0 1 0 0 0 1]
Accuracy: 0.975
[[55  3]
 [ 0 62]]
```

```
                   precision    recall  f1-score   support

                0       1.00      0.95      0.97        58
                1       0.95      1.00      0.98        62

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

```
# AdaBoostClassifier:
from sklearn.ensemble import AdaBoostClassifier
AdaBoost = AdaBoostClassifier()
AdaBoost = AdaBoost.fit(X_train,y_train)

# Predictions:
y_pred = AdaBoost.predict(X_test)
print(y_pred)
# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
     [1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0
      1 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0
      0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 0
```

Saved successfully!

```
      [ 0 62]]
                   precision    recall  f1-score   support

                0       1.00      0.95      0.97        58
                1       0.95      1.00      0.98        62

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

```
# GradientBoostingClassifier:
from sklearn.ensemble import GradientBoostingClassifier
GradientBoost = GradientBoostingClassifier()
GradientBoost = GradientBoost.fit(X_train,y_train)

# Predictions:
y_pred = GradientBoost.predict(X_test)
print(y_pred)
# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0
 1 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0
 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 0
 1 1 1 0 1 0 0 0 1]
Accuracy: 0.975
[[55  3]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        58
           1       0.95      1.00      0.98        62

                                       0.97       120
                                97     0.97       120
                                97     0.97       120
```

Saved successfully!

```
import pickle
pickle.dump(RandomForest, open("chronic_kidney_disease_prediction_model.pkl", 'wb'))
```

Colab paid products - Cancel contracts here

✓ 0s completed at 1:31 PM

Saved successfully!