

---

## HCIN 620 Lab 6 Course Project

In this project we are tasked with predicting the stages of Chronic Kidney Disease based on Glomerular Filtration Rate (GFR). Information for dataset is available in this link

<http://archive.ics.uci.edu/dataset/336/chronic+kidney+disease>

In order to succeed in this final you will be required to input most of the python code. Please review all the previous labs before you begin, and read the instructions carefully.

Rather than using a question/answer format, we have commented the code cells with the notation **#TO DO** This is a placeholder technique (a To Do list, so to speak) commonly used in machine learning. Complete each #TODO task requested of you.

Good Luck!

Notebook by Reza Afra, Ph.D. and Barbara Berkovich, Ph.D., M.A.

Revised by Thidarat Tinnakornsrisuphap, Ph.D.

### ✓ Step 1: Environment Setup


You have learned that in order to setup an environment for your project, you'd need to first import the libraries you need.

```
#TO DO 1. Import all the libraries.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
```

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, accuracy_score, confusion_mat
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.compose import ColumnTransformer
```

```
# Suppress pesky warnings
import warnings
warnings.filterwarnings("ignore")
```

```
#TO DO 2. Add a print command to acknowledge completion of the import.  
print('Imports Completed.')
```

 Imports Completed.

## ✓ Step 2: Data Cleaning

Upload the data file called data-lab-6-ckd-courseproject.csv data.

Read the csv file into a data frame, and use the name of the dataframe to print the first and last 5 rows.

```
#TO DO 3. Load the data (lab 6) into a pandas dataframe called "data"
```

```
data = pd.read_csv('./data-lab-6-ckd-courseproject.csv')
```

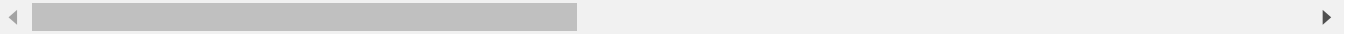
```
#TO DO 4. Use the name of the dataframe to print the first and last five rows.
```

```
data
```



	Age	Blood Pressure	Specific Gravity	Albumin	Sugar	Red Blood Cells	Pus Cell	Pus Cell clumps	Bacteria
0	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpres
1	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpres
2	63.0	70.0	1.010	3.0	0.0	abnormal	abnormal	present	notpres
3	68.0	80.0	1.010	3.0	2.0	normal	abnormal	present	pres
4	61.0	80.0	1.015	2.0	0.0	abnormal	abnormal	notpresent	notpres
...	...	...	...	...	...	...	...	...	...
153	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpres
154	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpres
155	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpres
156	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpres
157	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpres

158 rows × 25 columns



```
# TO DO 5. Perform data cleaning. How many missing values are in this dataset? Any fur
print(data.isnull().sum())
print('No NaNs found. No further action needed.')
```



Age	0
Blood Pressure	0
Specific Gravity	0
Albumin	0
Sugar	0
Red Blood Cells	0
Pus Cell	0
Pus Cell clumps	0
Bacteria	0
Blood Glucose Random	0
Blood Urea	0
Serum Creatinine	0
Sodium	0
Potassium	0
Hemoglobin	0
Packed Cell Volume	0
White Blood Cell Count	0
Red Blood Cell Count	0
Hypertension	0

```

Diabetes Mellitus      0
Coronary Artery Disease 0
Appetite               0
Pedal Edema           0
Anemia                0
Class                 0
dtype: int64
No NaNs found. No further action needed.

```

## ✓ Step 3: Exploratory Data Analysis (EDA) and Preprocessing

Next we're going to build the targets which are stages of CKD in a column we will call "CKD Stages" There are various equations for calculating GFR but here we will stick with a simplified form of it. Please read about GFR in the following link. Source:

<https://www.niddk.nih.gov/health-information/professionals/clinical-tools-patient-management/kidney-disease/laboratory-evaluation/glomerular-filtration-rate/estimating>

```

# Used a formula given by NIDDK which was simpler and made it even simpler by
# removing the effect of race and gender
# GFR (mL/min/1.73 m^2) = 175 × (Scr)^-1.154 × (Age)^-0.203

# DO NOT change the code below
def calc_gfr(Scr, Age): # Function to calculate GFR
    return (175 * (Scr) ** -1.154) * (Age ** -0.203) # GFR (mL/min/1.73 m^2) = 175 × (Sc

# I tried 6 stages but the dataset is too small so classes 4 and 5
# had no instances. I reduced the number of classes to 3.
def calc_ckd_stage(gfr): # Function for determining ckd stage.
    bins = [0, 45, 90, 250]
    labels = [3, 2, 1]
    ret = pd.cut(gfr, bins=bins, labels=labels)
    return ret

# DO NOT change the code below
data["GFR"] = calc_gfr(data["Serum Creatinine"], data["Age"])
gfr = data['GFR']
removed_outliers = gfr.between(gfr.quantile(.00), gfr.quantile(.95))

data = data[removed_outliers]

```

```
# DO NOT change the code below
data["CKD Stages"] =calc_ckd_stage(data["GFR"])
```

```
# DO NOT change the code below
data["CKD Stages"].value_counts()
```

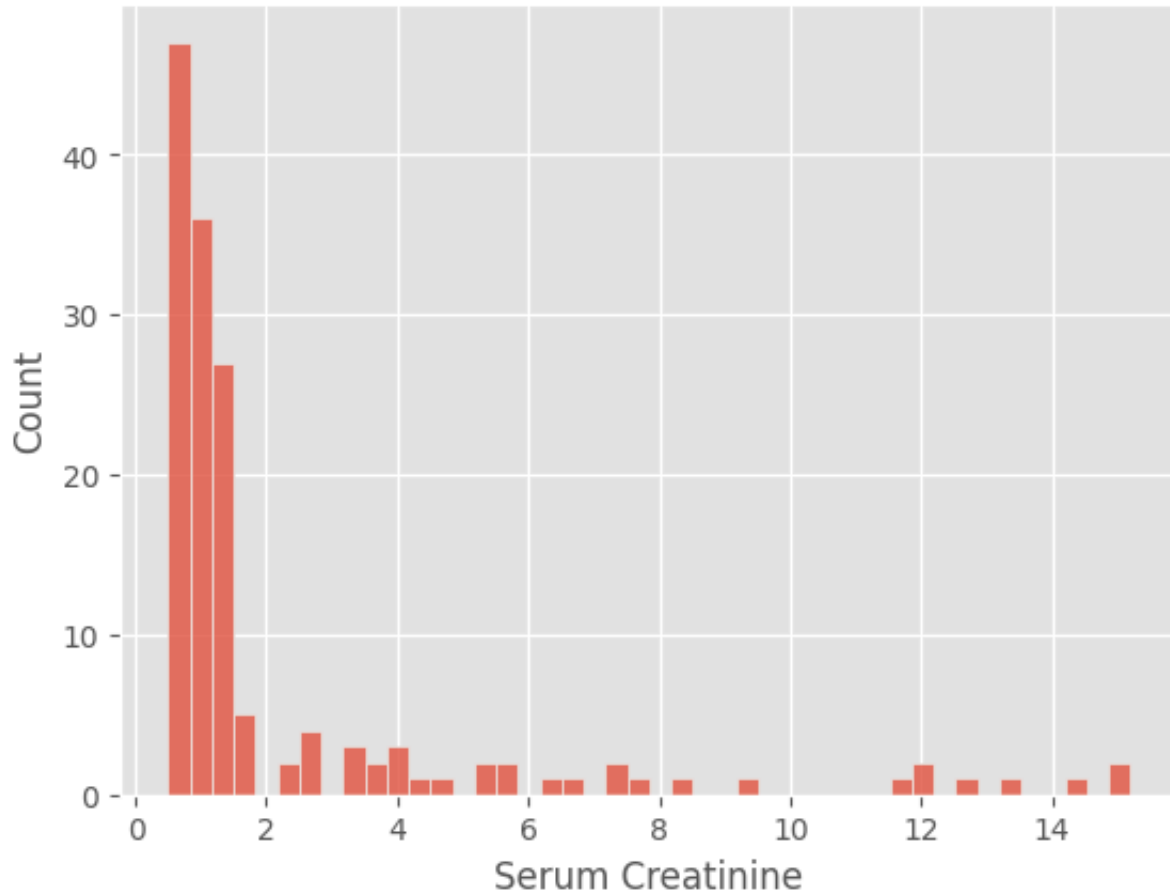


	count
CKD Stages	
1	58
2	53
3	39

**dtype:** int64

## ✓ Histogram

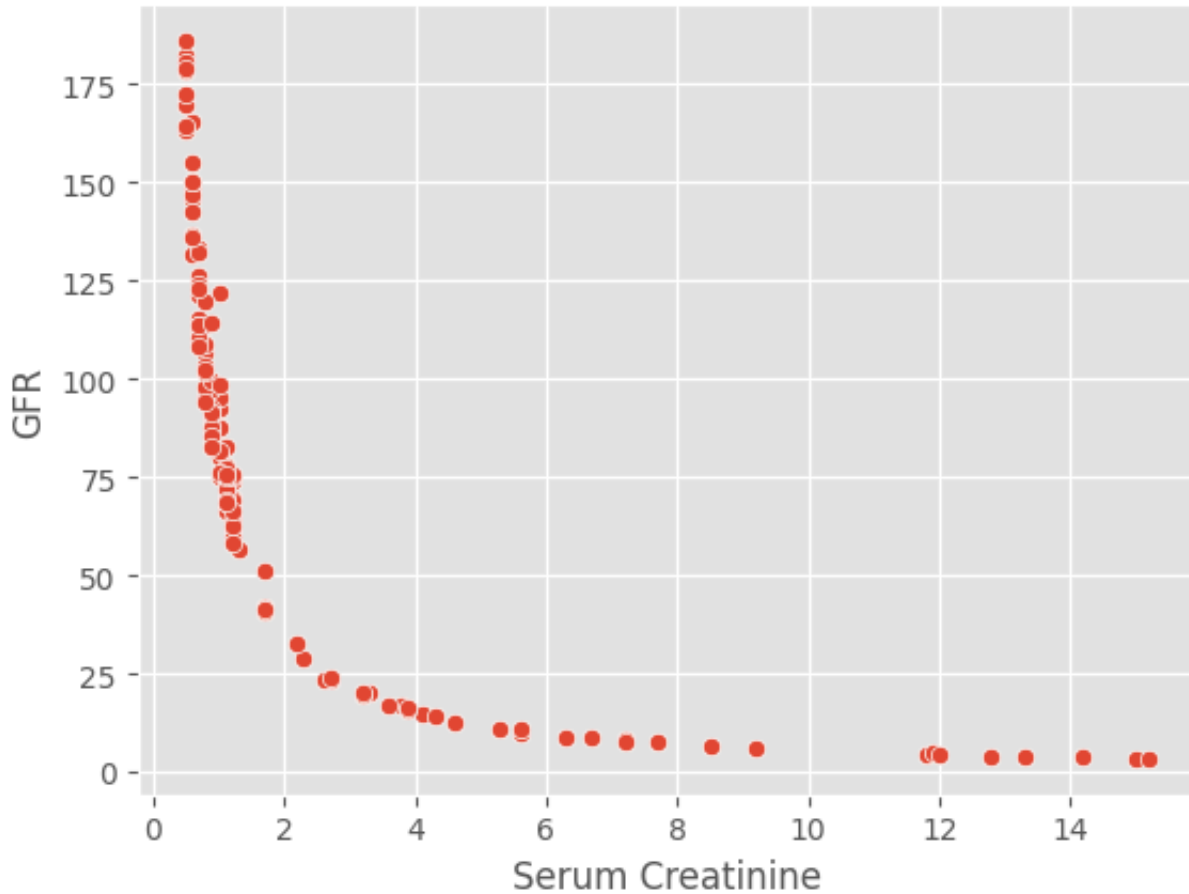
```
# TO DO 6: Create a histogram of the values of "Serum Creatinine" and provide interpre
sns.histplot(data, x='Serum Creatinine');
```



The majority of serum creatine levels are less than 2. This indicates healthy kidney function in the majority of patients.

## ✓ Scatterplot

```
# TO DO 7: Create a scatterplot of the values of "GFR" (y-axis) vs "Serum Creatinine"  
sns.scatterplot(data, x='Serum Creatinine', y='GFR');
```



There is an inversely proportional relationship between serum creatinine and GFR, as serum creatinine increases, GFR decreases exponentially.

## ✓ Isolate features from target

```
# TO DO 8: Isolate features and targets  
# DO NOT change the code below
```

```
X = data.drop(['Class', 'CKD Stages'], axis=1)  
y = data["CKD Stages"]
```

There are various ways to encode categorical data. You learned about some of them during labs. Visit this [link](#) and read it thoroughly. Find an appropriate encoding scheme and transform the categorical attributes of your dataset.

```
# TO DO 9: Isolate the categorical features and encode them using a proper encoding me  
# TO DO 10: Isolate the numerical features and scale them using a proper scaling methc
```

```
# DO NOT change the code below
```

```
numerical_ix = X.select_dtypes(include=['float64']).columns  
categorical_ix = X.select_dtypes(include=['object']).columns
```

```
#define the data preparation for the columns
```

```
t = [('cat', OneHotEncoder(), categorical_ix), ('num', StandardScaler(), numerical_ix)]  
transform = ColumnTransformer(transformers=t)
```

```
X = transform.fit_transform(X)
```

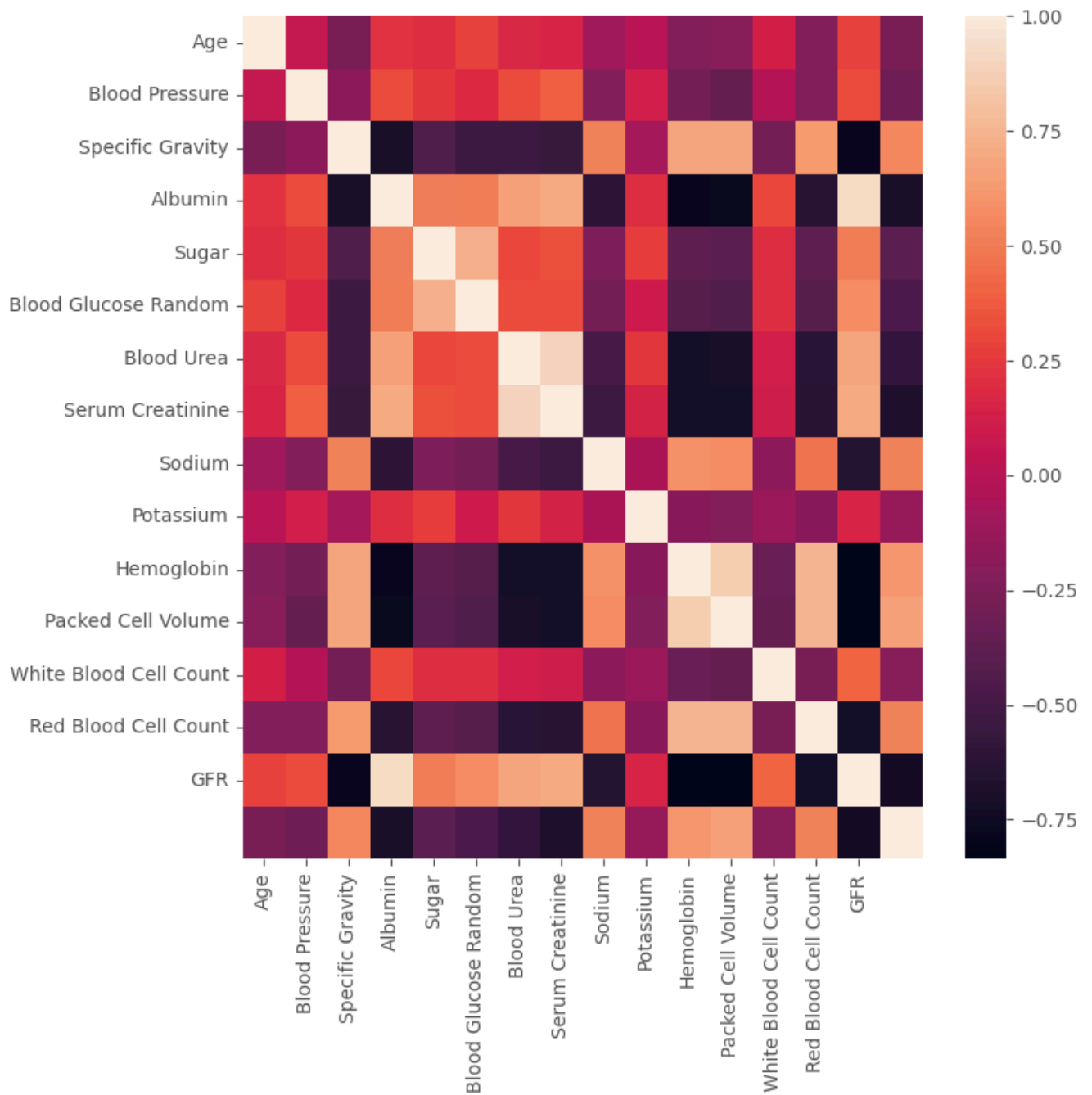
```
#TO DO 11: Create a heatmap of correlation between features. Pick at least 2 pairs of
```

```
# DO NOT change the code below
```

```
plt.figure(figsize=(8, 8))
```

```
sns.heatmap(data.corr(numeric_only = True), cbar=True, annot=False, yticklabels=numeri  
             cticklabels=numerical_ix);
```





There is a very nearly linear and positive relationship between Blood Urea and Serum Creatinine. This also is true of Hemoglobin and Packed Cell volume. The light color on both

indicates that the correlation between the two is very close to 1.

## ✓ Split the Data

```
# TO DO 12: Split the data to train and test sets
# DO NOT change the code below
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=data['CKD Stages'],
```

## ✓ Step 4: Build the Models and Evaluate

### ✓ Logistic Regression

```
# Use Logistic Regression to predict the stage of the kidney function
# and print the accuracies of your model on both train and test data
# DO NOT change the code below
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train) # Fitting the model with the training data
y_pred = log_reg.predict(X_test) # Prediction based on a value given from the test data
print(f' Accuracy on test set: {accuracy_score(y_test, y_pred):.3f}')
print(f' \nAccuracy on train set: {accuracy_score(y_train, log_reg.predict(X_train)):.3f}')
```

```
➡ Accuracy on test set: 1.000
```

```
Accuracy on train set: 0.991
```

### ✓ Confusion Matrix

```
# TO DO 13: Plot a Confusion Matrix based on your findings from the previous step
# DO NOT change the code below
```

```
data_ = {'y_true': y_test,
         'y_pred': y_pred
        }
```

```
df = pd.DataFrame(data_, columns=['y_true', 'y_pred'])
confusion_matrix = pd.crosstab(df['y_true'], df['y_pred'], rownames=['ACTUAL'], colnames=['PREDICTED'])
```

```
sns.heatmap(confusion_matrix, annot=True)
```

