Predict, Diagnose, and Treat Chronic Kidney Disease with Machine Learning

Pengxi Chen
McMaster University
April 19, 2024

[1]:
```
pip install ucimlrepo
```

```
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-info due to invalid metadata entry 'name'
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-info due to invalid metadata entry 'name'
Requirement already satisfied: ucimlrepo in /opt/conda/lib/python3.11/site-packages (0.0.6)
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-info due to invalid metadata entry 'name'
Note: you may need to restart the kernel to use updated packages.
```

[18]:
```
#(1)
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from ucimlrepo import fetch_ucirepo
from scipy import stats
from sklearn.ensemble import StackingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, f1_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder

chronic_kidney_disease = fetch_ucirepo(id=336)
```

```
# data
X = chronic_kidney_disease.data.features
y = chronic_kidney_disease.data.targets

#print(chronic_kidney_disease.metadata)
#print(chronic_kidney_disease.variables)
```

[19]:
```
#(2)
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns
scaler = MinMaxScaler()

X[numerical_cols] = scaler.fit_transform(X[numerical_cols])
print(X.head())
```

```
        age        bp    sg   al   su      rbc        pc          pcc  \
0  0.522727  0.230769  0.75  0.2  0.0      NaN    normal  notpresent
1  0.056818  0.000000  0.75  0.8  0.0      NaN    normal  notpresent
2  0.681818  0.230769  0.25  0.4  0.6   normal    normal  notpresent
3  0.522727  0.153846  0.00  0.8  0.0   normal  abnormal     present
4  0.556818  0.230769  0.25  0.4  0.0   normal    normal  notpresent

            ba       bgr  ...     hemo       pcv      wbcc      rbcc  htn  \
0  notpresent  0.211538  ...  0.836735  0.777778  0.231405  0.525424  yes
1  notpresent       NaN  ...  0.557823  0.644444  0.157025       NaN   no
2  notpresent  0.856838  ...  0.442177  0.488889  0.219008       NaN   no
3  notpresent  0.202991  ...  0.551020  0.511111  0.185950  0.305085  yes
4  notpresent  0.179487  ...  0.578231  0.577778  0.210744  0.423729   no

    dm  cad  appet   pe  ane
0  yes   no   good   no   no
1   no   no   good   no   no
2  yes   no   poor   no  yes
3   no   no   poor  yes  yes
4   no   no   good   no   no

[5 rows x 24 columns]
```

```
/tmp/ipykernel_192/71361484.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X[numerical_cols] = scaler.fit_transform(X[numerical_cols])
```

[20]:
```
#(3)
print(X.head())
print(X.dtypes)
```

```
print(X.describe())
for col in X.select_dtypes(include=['object']).columns:
    print(f"Unique values in {col}: {X[col].nunique()} - {X[col].unique()}")
print(y.value_counts(normalize=True))
```

```
        age        bp    sg   al   su     rbc        pc          pcc  \
0  0.522727  0.230769  0.75  0.2  0.0     NaN    normal  notpresent
1  0.056818  0.000000  0.75  0.8  0.0     NaN    normal  notpresent
2  0.681818  0.230769  0.25  0.4  0.6  normal    normal  notpresent
3  0.522727  0.153846  0.00  0.8  0.0  normal  abnormal     present
4  0.556818  0.230769  0.25  0.4  0.0  normal    normal  notpresent

            ba       bgr  ...     hemo       pcv      wbcc      rbcc htn  \
0  notpresent  0.211538  ...  0.836735  0.777778  0.231405  0.525424  yes
1  notpresent       NaN  ...  0.557823  0.644444  0.157025       NaN   no
2  notpresent  0.856838  ...  0.442177  0.488889  0.219008       NaN   no
3  notpresent  0.202991  ...  0.551020  0.511111  0.185950  0.305085  yes
4  notpresent  0.179487  ...  0.578231  0.577778  0.210744  0.423729   no

    dm  cad  appet   pe  ane
0  yes   no   good   no   no
1   no   no   good   no   no
2  yes   no   poor   no  yes
3   no   no   poor  yes  yes
4   no   no   good   no   no

[5 rows x 24 columns]
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
pot      float64
hemo     float64
pcv      float64
wbcc     float64
rbcc     float64
htn       object
dm        object
```

```
cad          object
appet        object
pe           object
ane          object
dtype: object
```

	age	bp	sg	al	su	bgr	\
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	
mean	0.562311	0.203608	0.620397	0.203390	0.090028	0.269309	
std	0.195110	0.105259	0.285831	0.270536	0.219838	0.169405	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.454545	0.153846	0.250000	0.000000	0.000000	0.164530	
50%	0.602273	0.230769	0.750000	0.000000	0.000000	0.211538	
75%	0.710227	0.230769	0.750000	0.400000	0.000000	0.301282	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

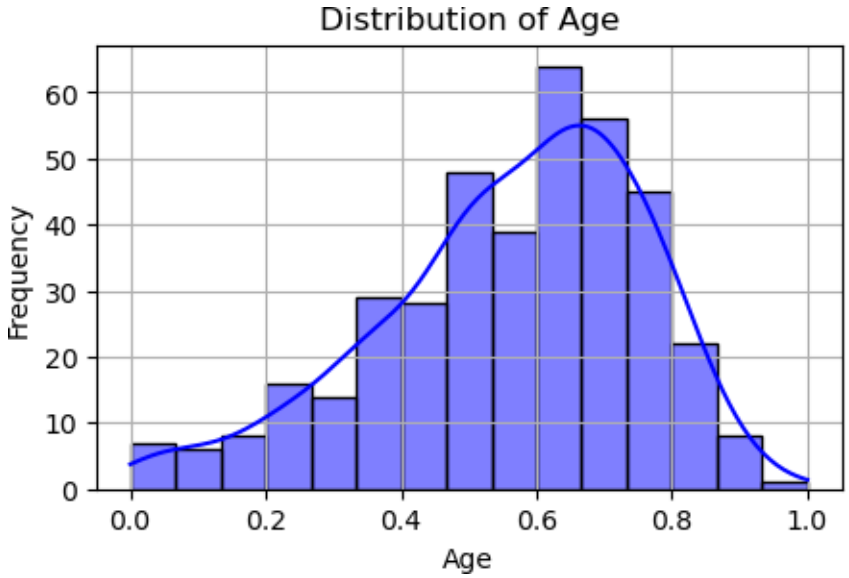
	bu	sc	sod	pot	hemo	pcv	\
count	381.000000	383.000000	313.000000	312.000000	348.000000	329.000000	
mean	0.143583	0.035350	0.839298	0.047803	0.641254	0.664100	
std	0.129661	0.075941	0.065670	0.071773	0.198135	0.199780	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.065469	0.006614	0.823344	0.029213	0.489796	0.511111	
50%	0.103979	0.011905	0.842271	0.042697	0.649660	0.688889	
75%	0.165597	0.031746	0.867508	0.053933	0.809524	0.800000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

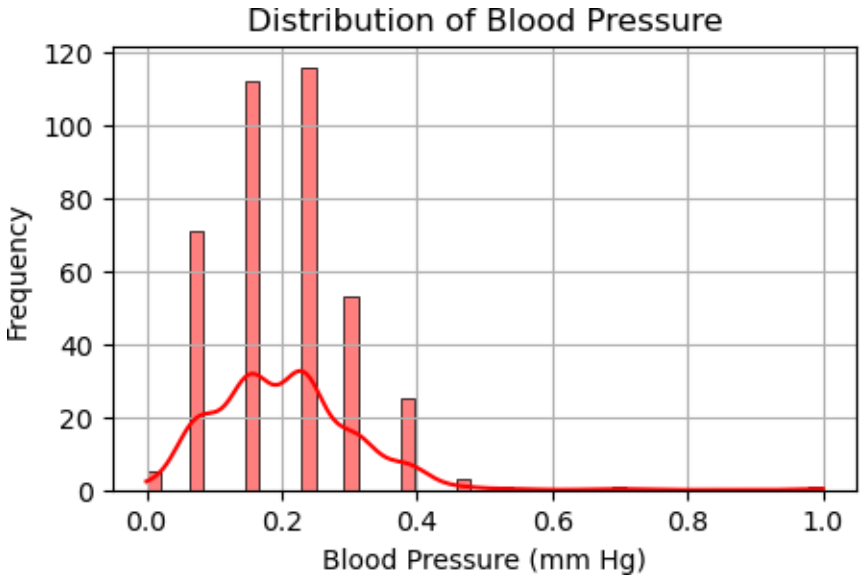
	wbcc	rbcc
count	294.000000	269.000000
mean	0.256451	0.441938
std	0.121672	0.173784
min	0.000000	0.000000
25%	0.177686	0.305085
50%	0.239669	0.457627
75%	0.314050	0.559322
max	1.000000	1.000000

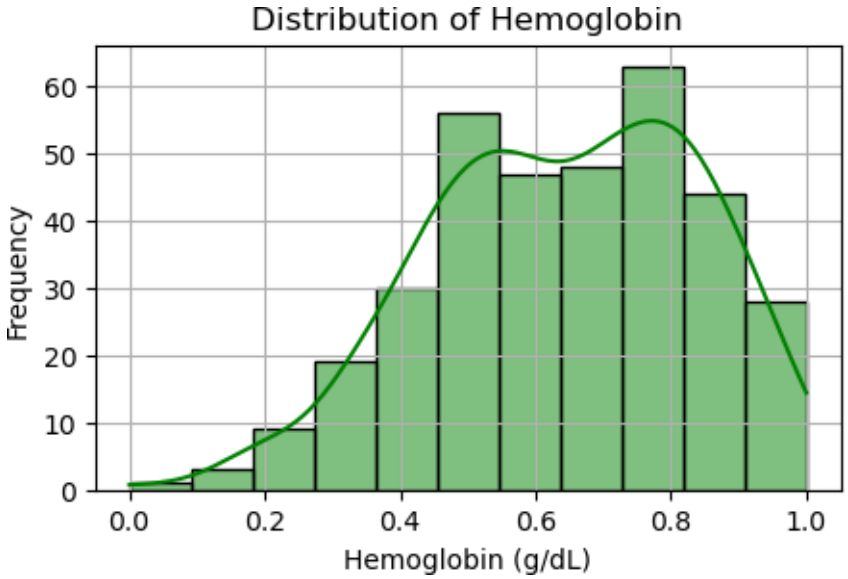
```
Unique values in rbc: 2 - [nan  'normal'  'abnormal']
Unique values in pc: 2 -  ['normal'  'abnormal'  nan]
Unique values in pcc: 2 - ['notpresent' 'present' nan]
Unique values in ba: 2 - ['notpresent' 'present' nan]
Unique values in htn: 2 - ['yes' 'no' nan]
Unique values in dm: 3 - ['yes' 'no' '\tno' nan]
Unique values in cad: 2 - ['no' 'yes' nan]
Unique values in appet: 2 - ['good' 'poor' nan]
Unique values in pe: 2 - ['no' 'yes' nan]
Unique values in ane: 2 - ['no' 'yes' nan]
class
ckd        0.620
notckd     0.375
ckd\t      0.005
```

dtype: float64

[21]:
```
#(3)
# Histogram for Age
plt.figure(figsize=(5, 3))
sns.histplot(X["age"], kde=True, color="blue")
plt.title("Distribution of Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

# Histogram for Blood Pressure
plt.figure(figsize=(5, 3))
sns.histplot(X["bp"], kde=True, color="red")
plt.title("Distribution of Blood Pressure")
plt.xlabel("Blood Pressure (mm Hg)")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

# Histogram for Hemoglobin plt.figure(figsize=(5,
3)) sns.histplot(X["hemo"], kde=True,
color="green") plt.title("Distribution of
Hemoglobin") plt.xlabel("Hemoglobin (g/dL)")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```



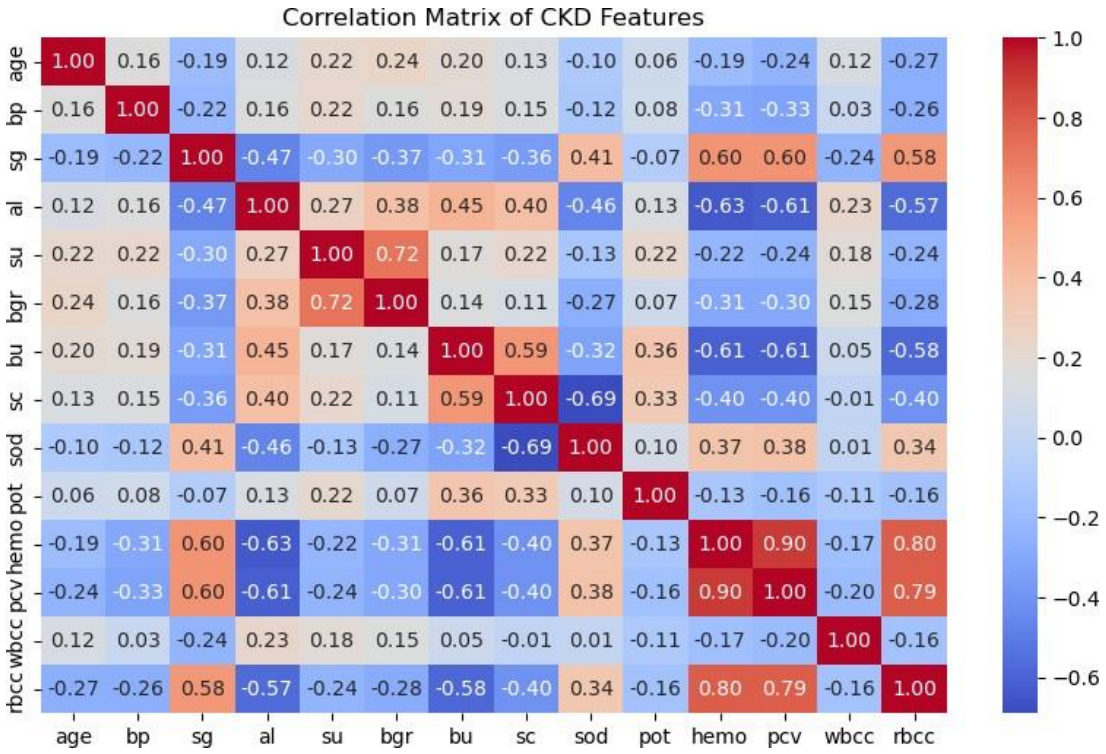




[22]:
```
#(4)
correlation_matrix = X.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Matrix of CKD Features")
```

```
plt.show()
```

```
/tmp/ipykernel_192/2860020853.py:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = X.corr()
```



[23]:
```
#(5)
numerical_features = X.select_dtypes(include=['int64', 'float64']).columns.
↪tolist()
categorical_features = X.select_dtypes(include=['object']).columns.tolist()
scaler = StandardScaler()
X[numerical_features] = scaler.fit_transform(X[numerical_features])
numerical_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean'))
])

categorical_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent'))
])
preprocessor = ColumnTransformer([
    ('num', numerical_pipeline, numerical_features),
```

```
    ('cat', categorical_pipeline, categorical_features)
])

X_transformed = preprocessor.fit_transform(X)
X_transformed_df = pd.DataFrame(X_transformed, columns=numerical_features +␣
↪categorical_features)

print(X_transformed_df.head())
```

```
        age        bp        sg        al        su       bgr        bu  \
0 -0.203139  0.258373  0.454071 -0.012548 -0.410106 -0.341498 -0.424804
1 -2.594124 -1.936857  0.454071  2.208413 -0.410106       0.0 -0.781687
2  0.613295  0.258373 -1.297699  0.727772  2.323069  3.473064 -0.087748
3 -0.203139 -0.47337 -2.173584  2.208413 -0.410106 -0.392022 -0.028268
4 -0.028189  0.258373 -1.297699  0.727772 -0.410106 -0.530963 -0.623073

         sc       sod       pot  ...      rbc        pc          pcc  \
0 -0.326574      -0.0      -0.0  ...   normal    normal   notpresent
1 -0.396338      -0.0      -0.0  ...   normal    normal   notpresent
2 -0.221928      -0.0      -0.0  ...   normal    normal   notpresent
3  0.126891 -2.552778 -0.667102  ...   normal  abnormal      present
4 -0.291692      -0.0      -0.0  ...   normal    normal   notpresent

            ba  htn   dm cad appet   pe  ane
0  notpresent  yes  yes  no  good   no   no
1  notpresent   no   no  no  good   no   no
2  notpresent   no  yes  no  poor   no  yes
3  notpresent  yes   no  no  poor  yes  yes
4  notpresent   no   no  no  good   no   no

[5 rows x 24 columns]
```

```
/tmp/ipykernel_192/4164833236.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X[numerical_features]    =    scaler.fit_transform(X[numerical_features])
```

[24]:
```
#(6)
for feature in numerical_features:
    Q1 = X_transformed_df[feature].quantile(0.25)
    Q3 = X_transformed_df[feature].quantile(0.75)
    IQR = Q3 - Q1
    outlier_step = 1.5 * IQR
    outliers = X_transformed_df[(X_transformed_df[feature] < Q1 - outlier_step)␣
↪| (X_transformed_df[feature] > Q3 + outlier_step )].index
```

```
    X_transformed_df = X_transformed_df.drop(outliers).reset_index(drop=True)

print(X_transformed_df.head())
```

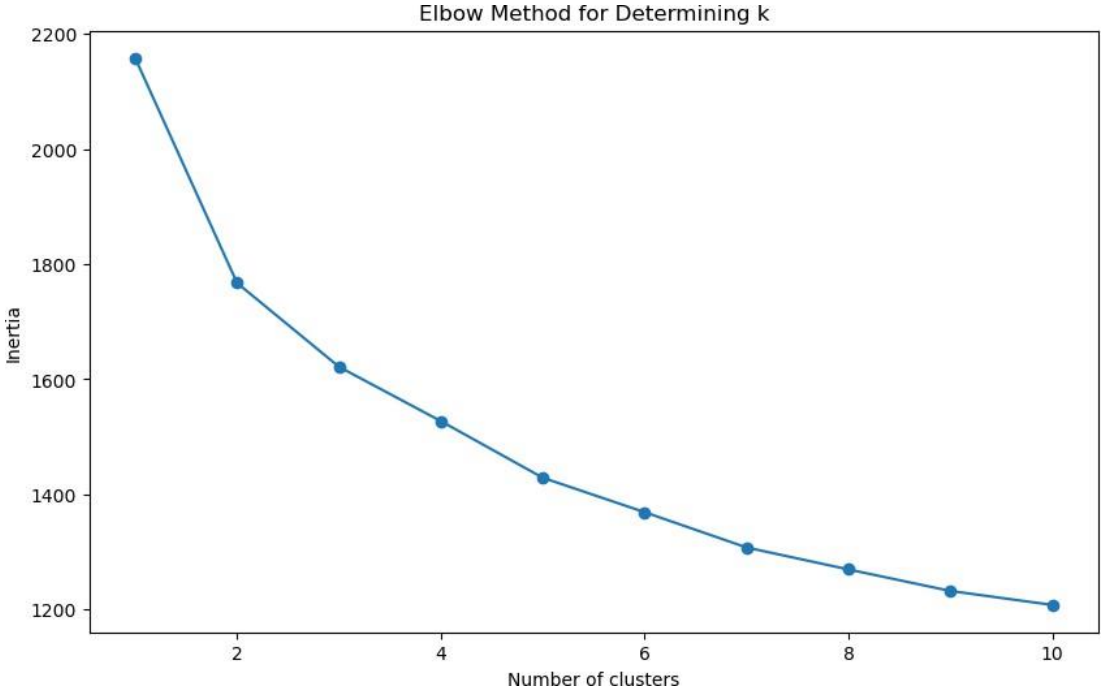
```
        age        bp        sg        al        su       bgr        bu  \
0 -0.203139  0.258373  0.454071 -0.012548 -0.410106 -0.341498 -0.424804
1 -0.028189  0.258373 -1.297699  0.727772 -0.410106 -0.530963 -0.623073
2  1.429729  -0.47337 -0.421814 -0.012548 -0.410106 -0.707797 -0.563592
3 -0.378089  -0.47337 -1.297699 -0.752868 -0.410106       0.0 -0.742034
4 -0.203139  -0.47337 -0.421814 -0.752868 -0.410106 -0.303605 -0.662726

         sc       sod       pot  ...      rbc      pc         pcc          ba  \
0 -0.326574      -0.0      -0.0  ...   normal  normal  notpresent  notpresent
1 -0.291692      -0.0      -0.0  ...   normal  normal  notpresent  notpresent
2 -0.221928 -0.435788 -0.228063  ...   normal  normal  notpresent  notpresent
3 -0.413779      -0.0      -0.0  ...   normal  normal  notpresent  notpresent
4 -0.326574  0.430254 -0.133983  ...   normal  normal  notpresent  notpresent

   htn   dm cad appet   pe ane
0  yes  yes  no  good   no  no
1   no   no  no  good   no  no
2  yes   no  no  good   no  no
3   no   no  no  good  yes  no
4   no  yes  no  good   no  no

[5 rows x 24 columns]
```

[25]:
```
#(7)
scaler = StandardScaler()
X_std = scaler.fit_transform(X_transformed_df[numerical_features])
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_std)
    inertia.append(kmeans.inertia_)

# Plot the elbow curve
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Determining k')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()

kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_std)
X_transformed_df['Cluster'] = clusters
```

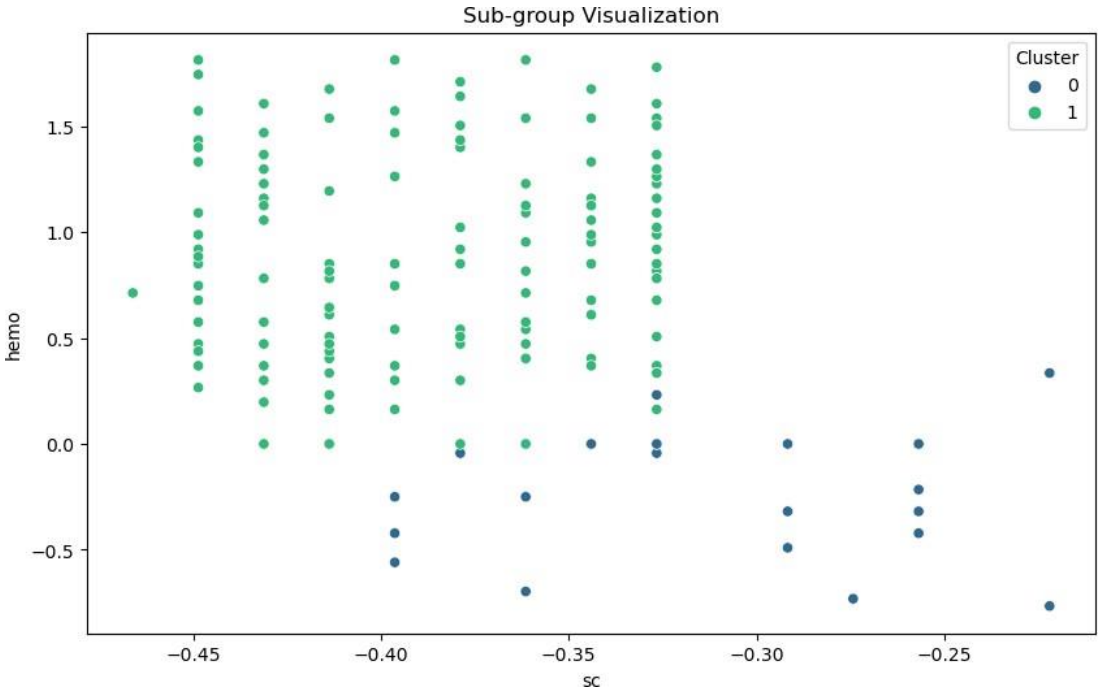
```python
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_transformed_df['sc'], y=X_transformed_df['hemo'],␣
↪hue=X_transformed_df['Cluster'], palette='viridis')
plt.title('Sub-group Visualization')
plt.xlabel('sc')
plt.ylabel('hemo')
plt.show()
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_transformed_df['bgr'], y=X_transformed_df['al'],␣
↪hue=X_transformed_df['Cluster'], palette='viridis')
plt.title('Sub-group Visualization')
plt.xlabel('bgr')
plt.ylabel('al')
plt.show()
```

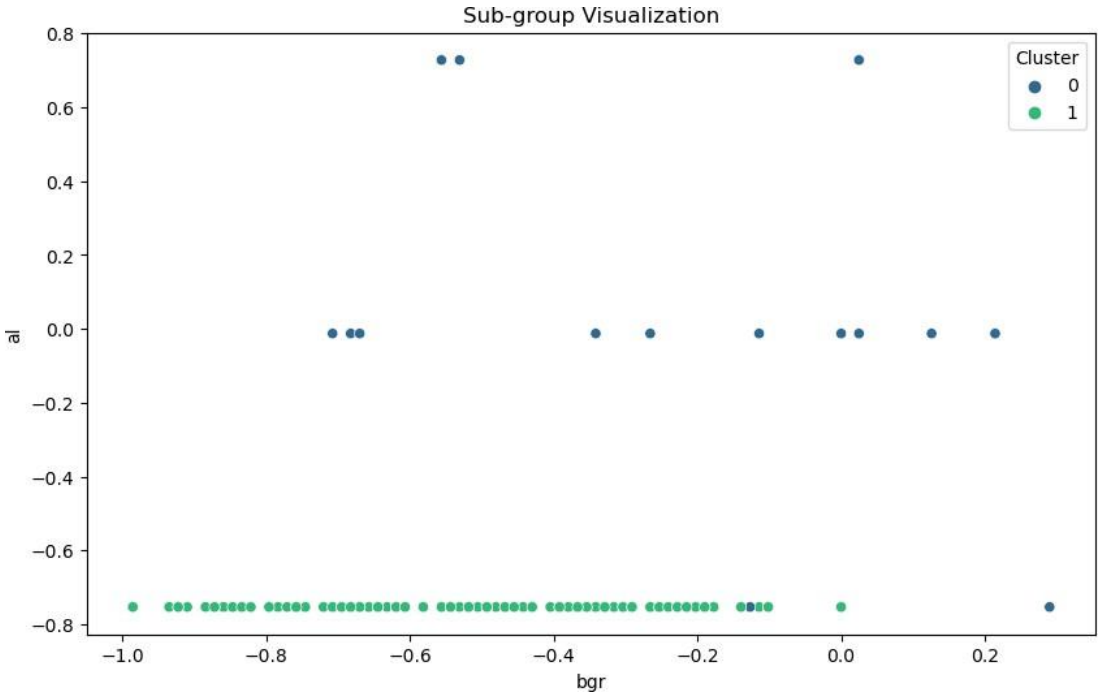
```
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
```

super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
super()._check_params_vs_input(X, default_n_init=10)
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
super()._check_params_vs_input(X, default_n_init=10)



/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
super()._check_params_vs_input(X, default_n_init=10)


Sub-group Visualization
Cluster
0
1
hemo
sc
1.5
1.0
0.5
0.0
−0.5
−0.45
−0.40
−0.35
−0.30
−0.25


Sub-group Visualization
Cluster
0
1
al
bgr
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2

[26]:
```
#(8)
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y,␣
↪test_size=0.3, random_state=1)

print("Training set size:", len(X_train))
print("Test set size:", len(X_test))
```

```
Training set size: 280
Test set size: 120
```

[38]:
```
#(11
for col in categorical_features:
    le = LabelEncoder()
    X[col]  =  le.fit_transform(X[col].astype(str))

pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

X_transformed = pipeline.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y,␣
↪test_size=0.3, random_state=1)

# Now you can fit the logistic regression model
logreg = LogisticRegression(random_state=1)
logreg.fit(X_train, y_train)
```

```
/opt/conda/lib/python3.11/site-packages/sklearn/utils/validation.py:1184:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[38]:
```
LogisticRegression(random_state=1)
```

[40]:
```
#(12)
rf = RandomForestClassifier(random_state=1)
logreg.fit(X_train,      y_train)
rf.fit(X_train, y_train)

# Predict probabilities for ROC AUC Score
logreg_probs = logreg.predict_proba(X_test)[:, 1]
rf_probs = rf.predict_proba(X_test)[:, 1]
```

```
# Predict class labels for F1 Score
logreg_preds = logreg.predict(X_test)
rf_preds = rf.predict(X_test)

# Calculate ROC AUC Score
logreg_roc_auc = roc_auc_score(y_test == 'ckd', logreg_probs)
rf_roc_auc = roc_auc_score(y_test == 'ckd', rf_probs)

# Calculate F1 Score
logreg_f1 = f1_score(y_test, logreg_preds, pos_label='ckd')
rf_f1 = f1_score(y_test, rf_preds, pos_label='ckd')

# Display the metrics
print(f"Logistic Regression - ROC AUC Score: {logreg_roc_auc}, F1 Score:
↪{logreg_f1}")
print(f"Random Forest - ROC AUC Score: {rf_roc_auc}, F1 Score: {rf_f1}")
```

```
/opt/conda/lib/python3.11/site-packages/sklearn/utils/validation.py:1184:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
/opt/conda/lib/python3.11/site-packages/sklearn/base.py:1151:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  return fit_method(estimator, *args, **kwargs)
Logistic Regression - ROC AUC Score: 0.7499999999999999, F1 Score:
0.9928057553956835
Random Forest - ROC AUC Score: 0.7020000000000001, F1 Score: 1.0
```

[41]:
```
#(13)

numerical_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object', 'category']).columns

numerical_transformer =
    Pipeline(steps=[ ('imputer',
    SimpleImputer(strategy='mean')), ('scaler',
    StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent'))
])

preprocessor =
```

```
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

clf =
    Pipeline(steps=[ ('preprocessor',
    preprocessor),
    ('classifier', LogisticRegression())
])


clf.fit(X, y.values.ravel())
logreg = clf.named_steps['classifier']

coefficients = logreg.coef_[0]
feature_names = numerical_features.tolist() + categorical_features.tolist()

coeff_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})
coeff_df = coeff_df.sort_values(by='Coefficient', ascending=False)
```

	Feature	Coefficient
22	pe	0.710421
3	al	0.494003
1	bp	0.481823
11	sc	0.446991
9	bgr	0.442852
16	wbcc	0.333391
10	bu	0.320817
8	ba	0.239194
4	su	0.207545
18	htn	0.205474
20	cad	0.205455
6	pc	0.147774
17	rbcc	0.047771
19	dm	-0.012258
13	pot	-0.022446
7	pcc	-0.139266
23	ane	-0.166385
21	appet	-0.206227
12	sod	-0.375719
0	age	-0.417736
15	pcv	-0.632986
2	sg	-0.718797
14	hemo	-0.735755
5	rbc	-1.011520

[42]:
```
#(14)

y_array = y.values.ravel()
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y_array,␣
↪test_size=0.3, random_state=1)

base_models = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=1)),
    ('logreg', LogisticRegression(random_state=1))
]

meta_model = LogisticRegression(random_state=1)

stacked_model = StackingClassifier(estimators=base_models,␣
↪final_estimator=meta_model, cv=5)

stacked_model.fit(X_train, y_train)

stacked_probs = stacked_model.predict_proba(X_test)[:, 1]
stacked_preds = stacked_model.predict(X_test)

stacked_roc_auc = roc_auc_score(y_test == 'ckd', stacked_probs)
stacked_f1 = f1_score(y_test, stacked_preds, pos_label='ckd')

print(f"Stacked Model - ROC AUC Score: {stacked_roc_auc}, F1 Score:␣
↪{stacked_f1}")
```

```
/opt/conda/lib/python3.11/site-packages/sklearn/model_selection/_split.py:725:
UserWarning: The least populated class in y has only 2 members, which is less
than n_splits=5.
  warnings.warn(
Stacked Model - ROC AUC Score: 0.9808571428571429, F1 Score: 0.9928057553956835
```