CHRONIC KIDNEY DISEASE CASE

-HARSH JAIN

Objective

- Identify the people who are at higher risk of CKD
- Using the given training dataset (6000 subjects) to identify important risk factors for CKD

Data Cleaning

- Create a new variable for Obese
- Obese variable has more null values
- While the variables, Height & Weight have lesser null values
- Assigning 0 & 1 values to dummy_Obese if BMI>=30

```
df.isnull().sum()
```

```
Racegrp          0
Educ            20
Unmarried      452
Income        1166
CareSource       0
Insured        113
Weight         194
Height         191
BMI            290
Obese          290
Waist          314
SBP            308
DBP            380
HDL             17
LDL             18
Total Chol      16
Dyslipidemia     0
PVD              0
Activity        10
PoorVision     567
```

```
df.describe()
df.shape
```

```
]: (8819, 34)
```

```
d_obese=np.where(df['Weight']/(df['Height']/100)**2 <30, 0,1)
df.insert(3, 'dummy_Obese', d_obese)
```

Data Cleaning

- Drop unnecessary columns
- Drop the rows with null values

```
df.drop(['ID','Educ','CareSource','Unmarried','Income','Insured','BMI','Obese','Weight',
       'Height','Waist','PoorVision'],inplace=True, axis=1)
```

```
df.shape
```

```
(8819, 23)
```

```
df.isna().sum()
```

```
Age                   0
Female                0
dummy_Obese           0
Racegrp               0
SBP                 308
DBP                 380
HDL                  17
LDL                  18
Total Chol           16
Dyslipidemia          0
PVD                   0
Activity             10
Smoker                0
Hypertension         80
Fam Hypertension      0
Diabetes              2
Fam Diabetes          0
Stroke               11
CVD                  23
Fam CVD             419
CHF                  36
Anemia                6
CKD                2819
```

```
df = df.dropna(subset=['SBP','DBP','HDL','LDL','Total Chol','Activity','Hypertension','Diabetes','Stroke','CVD','Fam CVD',
                       'CHF','Anemia'])
```

Data Cleaning

- Target Variable CKD has many NULL values
- Drop the rows with null values
- Size of the dataset after cleaning is (5381, 23)

```
df.isnull().sum()
```

```
Age                    0
Female                 0
dummy_Obese            0
Racegrp                0
SBP                    0
DBP                    0
HDL                    0
LDL                    0
Total Chol             0
Dyslipidemia           0
PVD                    0
Activity               0
Smoker                 0
Hypertension           0
Fam Hypertension       0
Diabetes               0
Fam Diabetes           0
Stroke                 0
CVD                    0
Fam CVD                0
CHF                    0
Anemia                 0
CKD                 2518
dtype: int64
```

```
df = df.dropna(subset=['CKD'])
```

```
df.shape
```

```
(5381, 23)
```

Data Cleaning

- All the variables are float or int type except Racegrp
- If Racegrp is White assign 1 else 0

```
df['Racegrp']=np.where(df['Racegrp'] !='white', 0,1)
```

```
df.dtypes
```

```
Age                    int64
Female                 int64
dummy_Obese            int32
Racegrp               object
SBP                  float64
DBP                  float64
HDL                  float64
LDL                  float64
Total Chol           float64
Dyslipidemia           int64
PVD                    int64
Activity             float64
Smoker                 int64
Hypertension         float64
Fam Hypertension       int64
Diabetes             float64
Fam Diabetes           int64
Stroke               float64
CVD                  float64
Fam CVD              float64
CHF                  float64
Anemia               float64
CKD                  float64
dtype: object
```

```
df['Racegrp'].unique()
```

```
array(['white', 'hispa', 'black', 'other'], dtype=object)
```

Exploratory Data Analysis

- Records have almost same no. of males and females (47%, 52%)
- Most of the data is about the people who are physically less active (mostly sitting, standing or walking)

```
df.Female.value_counts()
```

```
1    2826
0    2555
Name: Female, dtype: int64
```

```
df.Activity.value_counts()
```

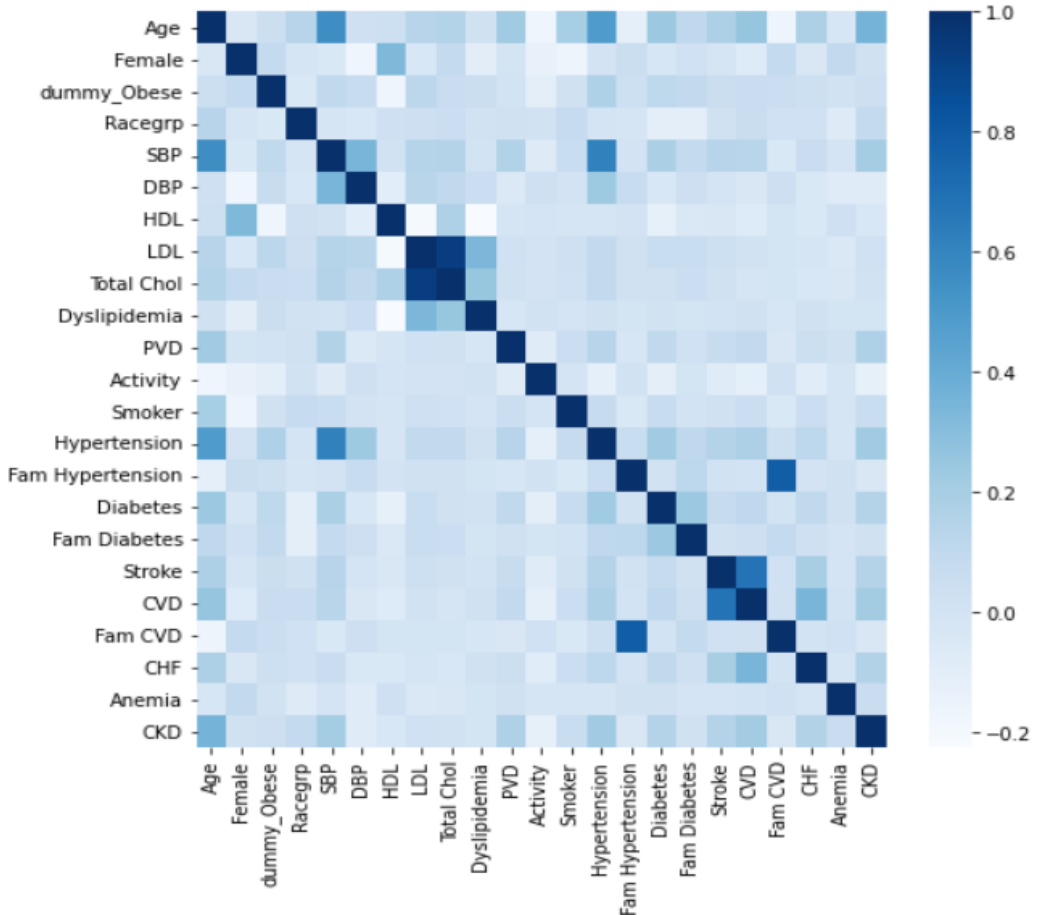
```
2.0    2863
1.0    1352
3.0     819
4.0     347
Name: Activity, dtype: int64
```

Exploratory Data Analysis

- Multivariate Correlation; darker the color higher the correlation between the variables

```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),cmap='Blues',annot=False)
```

```
<AxesSubplot:>
```



Exploratory Data Analysis

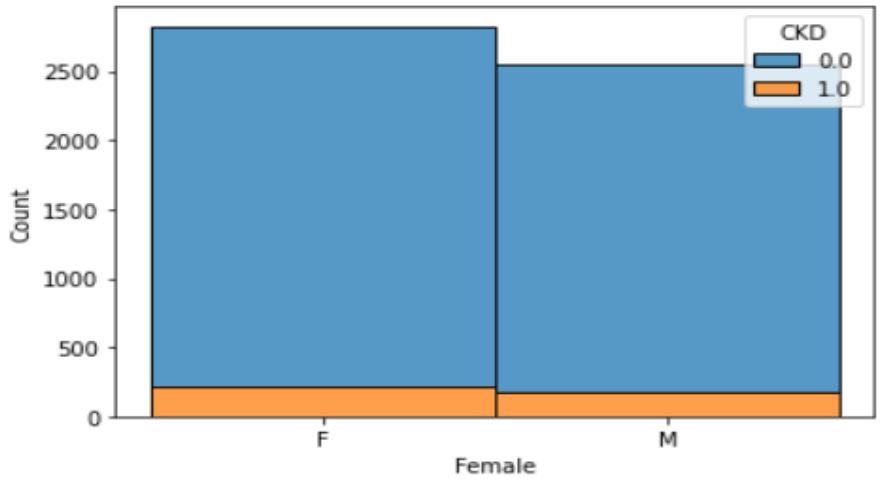
CKD in Male vs Female

```
a=df[['Female','CKD']]
a['Female']=np.where(a['Female']==0, 'M','F')
sns.histplot(binwidth=0.5, x="Female", hue="CKD", data=a, stat="count", multiple="stack")
```

```
C:\Users\hjain25\AppData\Local\Temp\ipykernel_21160\1051328237.py:2: 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_gui
-versus-a-copy
  a['Female']=np.where(a['Female']==0, 'M','F')
```

```
<AxesSubplot:xlabel='Female', ylabel='Count'>
```

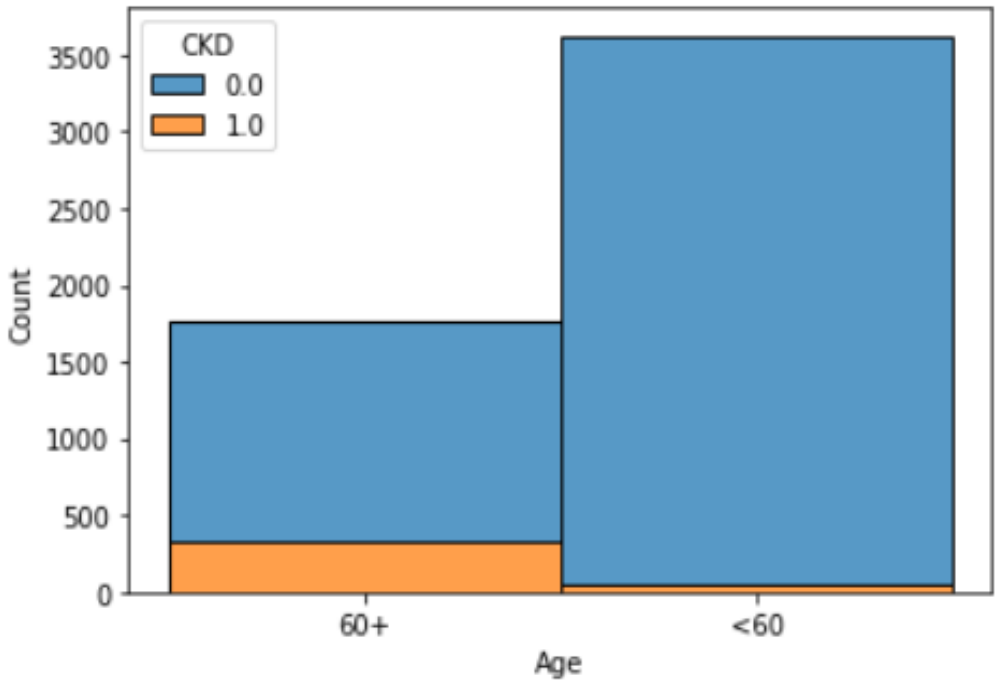


Exploratory Data Analysis

CKD in Age 60+

```
a=df[['Age','CKD']]
a['Age']=np.where(a['Age']>=60, '60+','<60')
sns.histplot(binwidth=0.5, x="Age", hue="CKD", data=a, stat="count", multiple="stack")
```

```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```

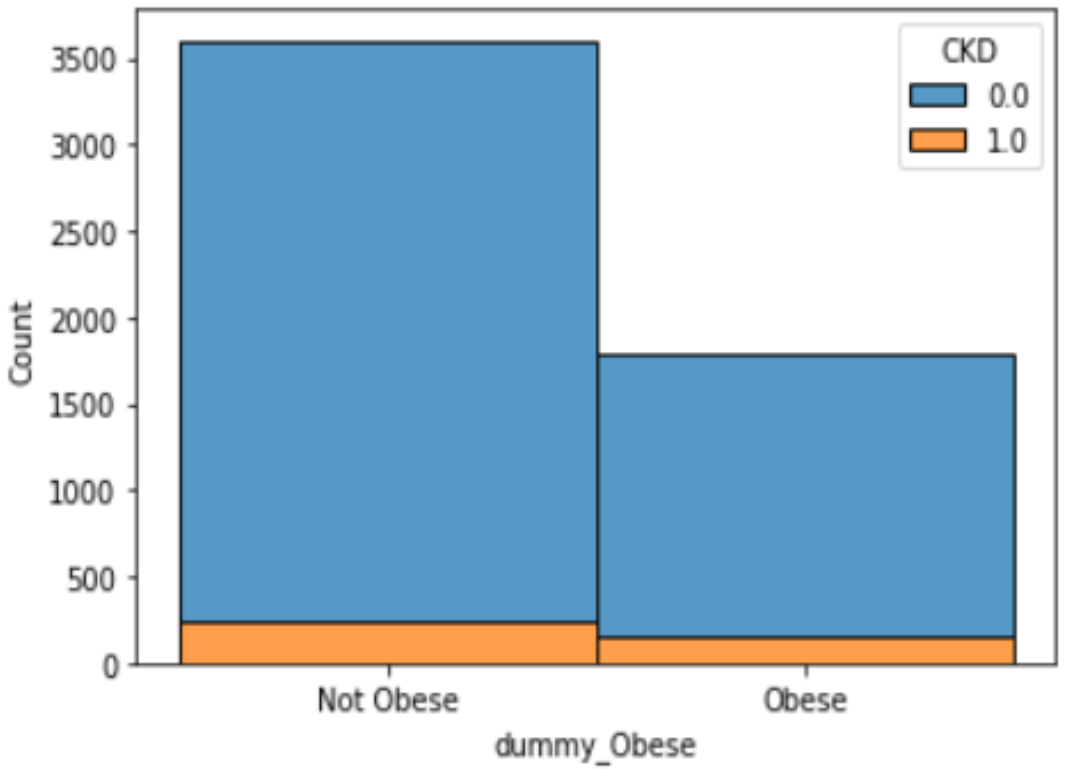


Exploratory Data Analysis

- CKD in Obese vs Not Obese
- As per the graph Not Obese people are not more likely to get CKD, as the number of observations are for Not Obese are double to that of the number of observations of Obese

```
a=df[['dummy_Obese','CKD']]
a['dummy_Obese']=np.where(a['dummy_Obese']==0, 'Not Obese','Obese')
sns.histplot(binwidth=0.5, x="dummy_Obese", hue="CKD", data=a, stat="count", multiple="stack")
```

```
<AxesSubplot:xlabel='dummy_Obese', ylabel='Count'>
```



PCA Analysis

```
from sklearn.decomposition import PCA
X = array[:,0:22]
Y = array[:,0:22]
# feature extraction
pca = PCA(n_components=3)
fit = pca.fit(X)
# summarize components
print("Explained Variance: %s" % fit.explained_variance_ratio_)
print(fit.components_)
```

```
Explained Variance: [0.731 0.126 0.078]
[[ 5.608e-02  1.952e-04  7.464e-04  3.872e-04  6.394e-02  2.794e-02
  -5.569e-03  7.070e-01  7.015e-01  1.563e-03  8.398e-05 -3.327e-05
   2.671e-04  1.011e-03  1.326e-04  2.585e-04  4.747e-04  9.808e-05
   1.654e-05 -7.097e-05 -4.820e-05 -1.032e-04]
 [ 6.070e-01 -1.113e-03  1.300e-03  8.646e-04  7.752e-01  1.450e-01
   3.288e-02 -7.849e-02 -4.562e-02 -6.736e-04  1.504e-03 -4.030e-03
   2.429e-03  1.259e-02 -8.398e-04  2.798e-03  1.825e-03  1.167e-03
   2.013e-03 -1.902e-03  7.922e-04 -9.000e-05]
 [ 3.881e-02  8.686e-03 -4.229e-03  9.929e-04 -5.898e-02 -1.149e-01
   8.089e-01 -3.962e-01  4.127e-01 -3.444e-03 -1.677e-04 -4.519e-04
  -1.955e-04 -1.163e-03 -2.192e-04 -1.847e-03 -1.193e-03 -3.141e-04
  -7.881e-04 -2.014e-04 -3.184e-04  2.268e-04]]
```

Feature Extraction & Ranking

```
from pandas import read_csv
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

array = df.values
X = array[:,0:22]
Y = array[:,22]
# feature extraction
model = LogisticRegression(solver='lbfgs')
rfe = RFE(model, step=3)
fit = rfe.fit(X, Y)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 11
Selected Features: [False  True  True  True False False False False False False  True  True
 False  True  True  True False False  True False  True  True]
Feature Ranking: [3 1 1 1 5 4 4 5 5 3 1 1 4 1 1 1 3 2 1 2 1 1]
```

Logistic Regression Model

- Accuracy of the model 93%

```
lr = linear_model.LogisticRegression()
```

```
data2= df.drop(['CKD'], axis=1)
x_train, x_test, y_train, y_test = train_test_split(data2, df['CKD'], test_size = 0.25)

lr = linear_model.LogisticRegression()
model = lr.fit(x_train,y_train)
y_pred = model.predict(x_test)
model.score(x_train,y_train)
```

```
C:\Users\hjain25\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_m
eWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
0.9301115241635688
```

Confusion Matrix

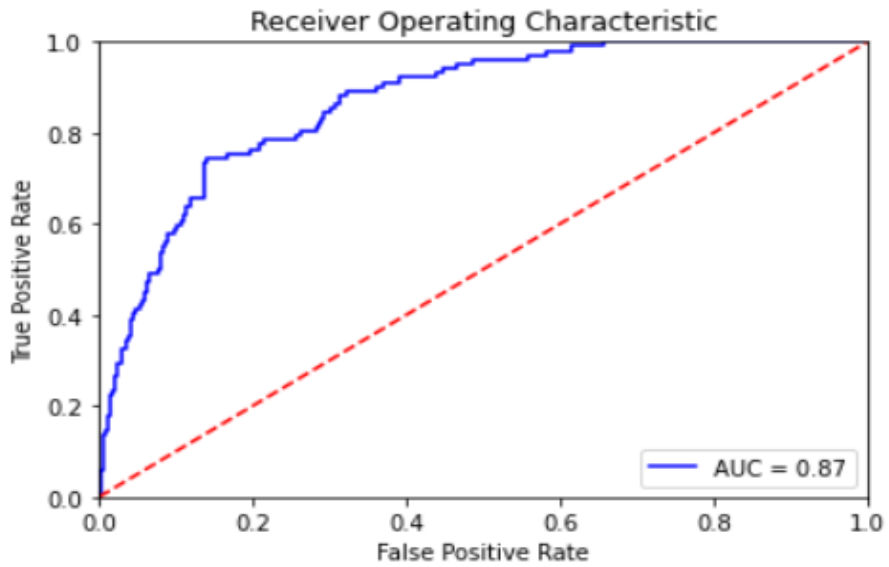
- The result is telling that we have 1227+20 correct predictions and 17+82 incorrect predictions

```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[1227   17]
 [  82   20]]
```

ROC Curve

- Another frequent tool used with binary classifiers is the (ROC) curve. The ROC curve of a random classifier is represented by the dotted line; a good classifier stays as far away from that line.
- The results of the area under the ROC curve (AUC) are considered excellent for AUC values of 0.9-1 and good for AUC values of 0.8-0.9.




```python
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = model.predict_proba(x_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Thank You!