# Data Collection and Preprocessing Phase

| | |
|---|---|
| Date | 12 July 2024 |
| Team ID | SWTID1720083491 |
| Project Title | Early Prediction of Chronic Kidney Disease Using Machine Learning |
| Maximum Marks | 6 Marks |

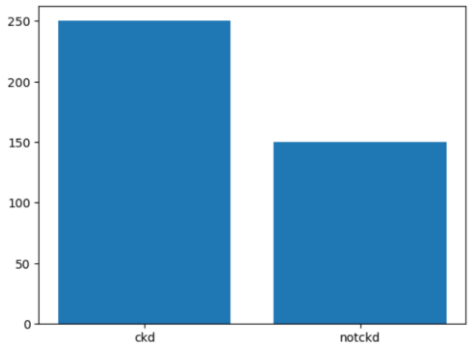
## Data Exploration and Preprocessing Template

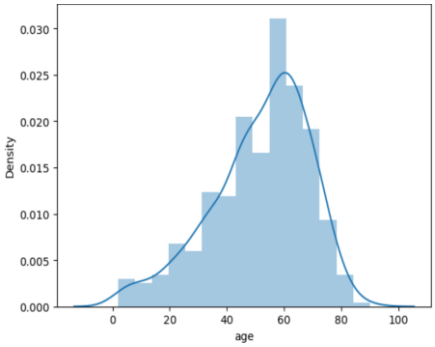
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

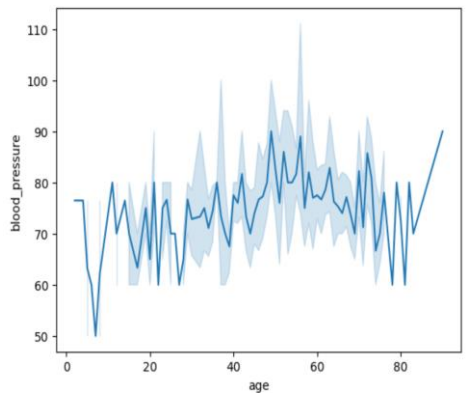
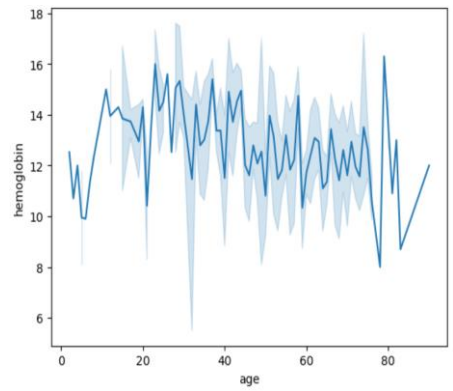
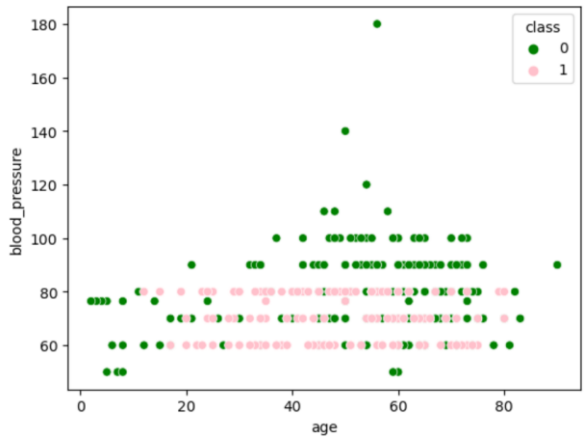
| Section | Description |
|---|---|
| Data Overview | <u>Dimension:</u><br>400 rows x 26 columns<br><u>Descriptive statistics:</u> |

| | age | blood_pressure | specific_gravity | albumin | sugar | blood_glucose_random | blood_urea | serum_creatinine | sodium | potassium | hemoglobin |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 391.000000 | 388.000000 | 353.000000 | 354.000000 | 351.000000 | 356.000000 | 381.000000 | 383.000000 | 313.000000 | 312.000000 | 348.000000 |
| mean | 51.483376 | 76.469072 | 1.017408 | 1.016949 | 0.450142 | 148.036517 | 57.425722 | 3.072454 | 137.528754 | 4.627244 | 12.526437 |
| std | 17.169714 | 13.683637 | 0.005717 | 1.352679 | 1.099191 | 79.281714 | 50.503006 | 5.741126 | 10.408752 | 3.193904 | 2.912587 |
| min | 2.000000 | 50.000000 | 1.005000 | 0.000000 | 0.000000 | 22.000000 | 1.500000 | 0.400000 | 4.500000 | 2.500000 | 3.100000 |
| 25% | 42.000000 | 70.000000 | 1.010000 | 0.000000 | 0.000000 | 99.000000 | 27.000000 | 0.900000 | 135.000000 | 3.800000 | 10.300000 |
| 50% | 55.000000 | 80.000000 | 1.020000 | 0.000000 | 0.000000 | 121.000000 | 42.000000 | 1.300000 | 138.000000 | 4.400000 | 12.650000 |
| 75% | 64.500000 | 80.000000 | 1.020000 | 2.000000 | 0.000000 | 163.000000 | 66.000000 | 2.800000 | 142.000000 | 4.900000 | 15.000000 |
| max | 90.000000 | 180.000000 | 1.025000 | 5.000000 | 5.000000 | 490.000000 | 391.000000 | 76.000000 | 163.000000 | 47.000000 | 17.800000 |

Univariate Analysis



| | |
|---|---|
| Bivariate Analysis |     |
| Multivariate Analysis |   |
| Outliers and Anomalies | ```python<br>import pandas as pd<br>import numpy as np<br><br>def remove_outliers_with_median(fd, cols_to_check, threshold=3.5):<br><br>  for col in cols_to_check:<br>  if pd.api.types.is_numeric_dtype(fd[col]):  # Check for numeric data type<br>  # Calculate z-scores efficiently using vectorized operations<br>  z_scores = np.abs(stats.zscore(fd[col]))<br><br>  # Identify outliers based on z-score threshold<br>  outlier_indices = np.where(z_scores > threshold)[0]<br><br>  # Replace outliers with median in one step<br>  fd.loc[outlier_indices, col] = fd[col].median()<br><br>  return fd<br><br>cols_to_check = ["bgr", "bu", "wc", "sc", "bp"]  # Your continuous columns<br>df_without_outliers = remove_outliers_with_median(df1.copy(), cols_to_check)<br><br>df_without_outliers.isna().sum()<br>```  |

**Data Preprocessing Code Screenshots**

## Loading Data

```
#2. Read the Dataset
df = pd.read_csv('chronickidneydisease.csv')
df
```

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

## Handling Missing Data

```
# Handling the Null Values
# with mean()
df['blood_glucose_random'].fillna(df['blood_glucose_random'].mean(),inplace=True)
df['blood_pressure'].fillna(df['blood_pressure'].mean(),inplace=True)
df['blood_urea'].fillna(df['blood_urea'].mean(),inplace=True)
df['hemoglobin'].fillna(df['hemoglobin'].mean(),inplace=True)
df['packed_cell_volume'].fillna(df['packed_cell_volume'].mean(),inplace=True)
df['potassium'].fillna(df['potassium'].mean(),inplace=True)
df['red_blood_cell_count'].fillna(df['red_blood_cell_count'].mean(),inplace=True)
df['serum_creatinine'].fillna(df['serum_creatinine'].mean(),inplace=True)
df['sodium'].fillna(df['sodium'].mean(),inplace=True)
df['white_blood_cell_count'].fillna(df['white_blood_cell_count'].mean(),inplace=True)
# with mode()
df['age'].fillna(df['age'].mode()[0],inplace=True)
df['hypertension'].fillna(df['hypertension'].mode()[0],inplace=True)
df['pus_cell_clumps'].fillna(df['pus_cell_clumps'].mode()[0],inplace=True)
df['appetite'].fillna(df['appetite'].mode()[0],inplace=True)
df['albumin'].fillna(df['albumin'].mode()[0],inplace=True)
df['pus_cell'].fillna(df['pus_cell'].mode()[0],inplace=True)
df['red_blood_cell'].fillna(df['red_blood_cell'].mode()[0],inplace=True)
df['coronary_artery_disease'].fillna(df['coronary_artery_disease'].mode()[0],inplace=True)
df['bacteria'].fillna(df['bacteria'].mode()[0],inplace=True)
df['anemia'].fillna(df['anemia'].mode()[0],inplace=True)
df['sugar'].fillna(df['sugar'].mode()[0],inplace=True)
df['diabetesmellitus'].fillna(df['diabetesmellitus'].mode()[0],inplace=True)
df['pedal_edema'].fillna(df['pedal_edema'].mode()[0],inplace=True)
df['specific_gravity'].fillna(df['specific_gravity'].mode()[0],inplace=True)
```

## Data Transformation

```
# Label Encoding
for i in catcols:
    print("Label Encoding of:",i)
    le= LabelEncoder()
    print(c(df[i]))
    df[i]=le.fit_transform(df[i])
    print(c(df[i]))
    print("*"*100)
```

```
# Scaling - MinMax Scaler
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
```

```
X_scaled=scale.fit_transform(x)
X_scaled
```

```
array([[1.        , 1.        , 0.21153846, ..., 0.        , 1.        ,
        0.        ],
       [1.        , 1.        , 0.2693088 , ..., 0.        , 0.        ,
        0.        ],
       [1.        , 1.        , 0.85683761, ..., 1.        , 1.        ,
        0.        ],
       ...,
       [1.        , 1.        , 0.16666667, ..., 0.        , 0.        ,
        0.        ],
       [1.        , 1.        , 0.1965812 , ..., 0.        , 0.        ,
        0.        ],
       [1.        , 1.        , 0.23290598, ..., 0.        , 0.        ,
        0.        ]])
```

Feature Engineering

```
#Rectifying the unknown class "ckd\t"
df['class']=df['class'].replace("ckd\t","ckd")
df['class'].unique()
```

```
# Removing the columns which are non Numerical
contcols.remove('specific_gravity')
contcols.remove('albumin')
contcols.remove('sugar')
print(contcols)
```

```
# Adding columns which we found Continuous
contcols.add('red_blood_cell_count')
contcols.add('packed_cell_volume')
contcols.add('white_blood_cell_count')
print(contcols)
```

```
# Rectifying the Categoirical columns Classes
df['coronary_artery_disease'] = df.coronary_artery_disease.replace('\tno','no')
c(df['coronary_artery_disease'])
```

```
Counter({'no': 364, 'yes': 34, nan: 2})
```

```
df['diabetesmellitus'] = df.diabetesmellitus.replace(to_replace={'\tno':'no','\tyes':'yes',' yes':'yes'})
c(df['diabetesmellitus'])
```

```
Counter({'no': 261, 'yes': 137, nan: 2})
```

| | |
|---|---|
| Save Processed Data | |

```python
# Choose a filename for the CSV (replace 'your_file.csv' with your desired name)
filename = 'Final_Data.csv'

# Export the DataFrame to a CSV file
df_filtered.to_csv(filename, index=False)  # Optional: set index=False to exclude the row index from the CSV

print(f'DataFrame exported to CSV file: {filename}')
```