Double-click (or enter) to edit

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
import plotly.express as px
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', '{:.2f}'.format)
```

```
df=pd.read_csv('/content/drive/MyDrive/FE/processed2.3.csv')
```

```
<ipython-input-29-535cd74af096>:1: DtypeWarning: Columns (41,51,53) have mi
  df=pd.read_csv('/content/drive/MyDrive/FE/processed2.3.csv')
```

```
df.dtypes
```

	0
RESEARCH_ID	object
SAMPLE_ID	object
COLLECTYEAR	int64
REGN_DATE	object
GENDER_NAME	object
AGE_YEARS	float64
AGE_DAYS	int64
AGE_MONTHS	int64
CITY_NAME	object
HEIGHT	int64
WEIGHT	float64
BMI	float64
Thyroid Stimulating Hormone (TSH)	float64
Uric Acid in Serum	float64
Alanine Aminotransferase (ALT)	float64
Ferritin In Serum	float64
Blood Urea Nitrogen (BUN)	float64
Lymphocytes absolute count	float64
R. B. Cs / HPFs	float64
Aspect(Urine Physical Examination) Ordinal Encoding	float64
Eosinophils absolute count	float64
Vitamin D (25 OH-Vit D -Total)	float64
C-Reactive Protein (CRP) quantitative	float64
Transferrin	float64
Red cell count	float64
Basophils absolute count	float64
Crystals(Urine Microscopic Examination :)	float64
Protein(Urine Physical Examination)	float64
Colour(Urine Physical Examination)	float64

Colour(Urine Physical Examination)	float64
Nitrite	float64
LDL Cholesterol	int64
LDL / HDL	float64
24 Hour Urine Volume (263)	float64
Hemoglobin	float64
Total Leucocytic Count	float64
Hematocrit	float64
MCV	float64
Glucose(Urine Physical Examination)	float64
Urea in Serum	float64
Prostatic Specific Antigen (PSA) Total	float64
Testosterone (Total)	float64
Alkaline Phosphatase	object
Total Protein in Serum	float64
Estimated Glomerular Filtration Rate(eGFR)	float64
Anti CCP Abs	int64
BUN/Creatinine Ratio	float64
Ketones	float64
MCHC	float64
pH(Urine Physical Examination)	float64
Amorphous Elements	float64
Blood and Haemoglobin	float64
Epithelial Cells / HPF	object
Casts(Urine Microscopic Examination :)	int64
Bilirubin	object
Chloride in Serum	float64
Cholesterol	float64
T. Cholesterol/HDL	float64
Urobilinogen	float64

R.B.Cs / HPF	float64
Erythrocyte Sedimentation Rate(ESR)	float64
Glucose in Plasma (Fasting)	float64
Hb A1c %	float64
Mean of blood glucose	float64
Microalbuminuria (24 h urine)	float64
Bilirubin (Total)	float64
Florescence Pattern	int64
Lead in blood	float64
Monocytes absolute count	float64
Consistancy	float64
Neutrophils absolute count	float64
Specific Gravity	float64
W. B. Cs / HPF	float64
Aspartate Aminotransferase (AST)	float64
Calcium in Serum (Total)	float64
Free T4	float64
Potassium (K) in Serum	float64
Albumin in Serum	float64
Iron (Fe) in Serum	float64
CRP H.S	float64
Triglycerides (TG) in Serum	float64
Rheumatoid Factor (quantitative)	float64
Platelet Count	float64
Albumin in Urine (263)	float64
MCH	float64
RDW	float64
W.B.Cs / HPF	float64
Leucocyte esterase	float64
Concentration	float64

Creatinine in Serum	float64
Sodium (Na) in Serum	float64
Bilirubin (Direct)	float64
Magnesium (Mg) in Serum	float64
Titre on Hep2 cells	float64
HDL Cholesterol	float64
Globulin in Serum	float64
Cystatin C	float64
RESEARCH_ID_int	int64
GENDER_BINARY	int64
CITY_NAME_ENCODED	int64
BDL	int64
Florescence Pattern	int64
Systolic Pressure	float64
Diastolic Pressure	float64
Amorphous_Numeric	float64
Bilirubin_Numeric	float64
T. Cholesterol/HDL_Numeric	float64

dtype: object

```
new_df = df[['AGE_YEARS', 'GENDER_NAME', 'WEIGHT', 'BMI', 'AGE_DAYS']].copy()
new_df.to_csv('AGE_YEARS_GENDER_NAME_WEIGH_BMI_AGE_DAYS', index=False)
```

```
label_encoders = {}
for col in ["GENDER_NAME", "CITY_NAME"]:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))
    label_encoders[col] = le
```

```
valid_df = df[(df["HEIGHT"].between(50, 250)) & (df["WEIGHT"].between(3, 300))]
```

```
train_data = valid_df.dropna(subset=["HEIGHT", "WEIGHT", "BMI"])
predict_data = df[(df["HEIGHT"] < 50) | (df["HEIGHT"] > 250) |
                  (df["WEIGHT"] < 3) | (df["WEIGHT"] > 300)]
```

```
features = ["AGE_YEARS", "GENDER_NAME", "CITY_NAME"]
X_train = train_data[features]
y_train_height = train_data["HEIGHT"]
y_train_weight = train_data["WEIGHT"]
```

```
model_height = RandomForestRegressor(n_estimators=100, random_state=42)
model_weight = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
model_height.fit(X_train, y_train_height)
model_weight.fit(X_train, y_train_weight)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
X_predict = predict_data[features]
```

```
df.loc[df["HEIGHT"] < 50, "HEIGHT"] = model_height.predict(X_predict[df["HEIGHT
df.loc[df["HEIGHT"] > 250, "HEIGHT"] = model_height.predict(X_predict[df["HEIGH
df.loc[df["WEIGHT"] < 3, "WEIGHT"] = model_weight.predict(X_predict[df["WEIGHT"
df.loc[df["WEIGHT"] > 300, "WEIGHT"] = model_weight.predict(X_predict[df["WEIGH
```

```
<ipython-input-39-00e1c7c68cfe>:1: UserWarning: Boolean Series key will be
  df.loc[df["HEIGHT"] < 50, "HEIGHT"] = model_height.predict(X_predict[df["
<ipython-input-39-00e1c7c68cfe>:1: FutureWarning: Setting an item of incomp
 168.9589672 ]' has dtype incompatible with int64, please explicitly cast t
  df.loc[df["HEIGHT"] < 50, "HEIGHT"] = model_height.predict(X_predict[df["
<ipython-input-39-00e1c7c68cfe>:2: UserWarning: Boolean Series key will be
  df.loc[df["HEIGHT"] > 250, "HEIGHT"] = model_height.predict(X_predict[df[
<ipython-input-39-00e1c7c68cfe>:3: UserWarning: Boolean Series key will be
  df.loc[df["WEIGHT"] < 3, "WEIGHT"] = model_weight.predict(X_predict[df["W
<ipython-input-39-00e1c7c68cfe>:4: UserWarning: Boolean Series key will be
  df.loc[df["WEIGHT"] > 300, "WEIGHT"] = model_weight.predict(X_predict[df[
```

```
df["HEIGHT"] = df["HEIGHT"].round().astype(int)
df["WEIGHT"] = df["WEIGHT"].round(2)
df["BMI"] = df["BMI"].clip(lower=15, upper=50).round(2)
```

```
df["BMI"] = df["WEIGHT"] / ((df["HEIGHT"] / 100) ** 2)
```

```
df["HEIGHT"].isna().sum()
```

```
np.int64(0)
```

```
df["BMI"].isna().sum()
```

```
np.int64(0)
```

```
df["WEIGHT"].isna().sum()
```

```
np.int64(0)
```

```
df["HEIGHT"] = df["HEIGHT"].round().astype(int)
df["WEIGHT"] = df["WEIGHT"].round(2)
df["BMI"] = df["BMI"].clip(lower=15, upper=50).round(2)
```

```
df.duplicated().sum()
```

```
np.int64(85)
```

```
df.drop_duplicates(inplace=True)
```

```
df.shape
```

```
(899915, 106)
```

```
all_columns = df.columns.tolist()
all_columns
```

```
['RESEARCH_ID',
 'SAMPLE_ID',
 'COLLECTYEAR',
 'REGN_DATE',
 'GENDER_NAME',
 'AGE_YEARS',
 'AGE_DAYS',
 'AGE_MONTHS',
 'CITY_NAME',
 'HEIGHT',
 'WEIGHT',
 'BMI',
 'Thyroid Stimulating Hormone (TSH)',
 'Uric Acid in Serum',
 'Alanine Aminotransferase (ALT)',
 'Ferritin In Serum',
 'Blood Urea Nitrogen (BUN)',
 'Lymphocytes absolute count',
 'R. B. Cs / HPFs',
 'Aspect(Urine Physical Examination) Ordinal Encoding',
 'Eosinophils absolute count',
 'Vitamin D (25 OH-Vit D -Total)',
 'C-Reactive Protein (CRP) quantitative',
 'Transferrin',
 'Red cell count',
 'Basophils absolute count',
 'Crystals(Urine Microscopic Examination :)',
 'Protein(Urine Physical Examination)',
 'Colour(Urine Physical Examination)',
 'Nitrite',
 'LDL Cholesterol',
 'LDL / HDL',
 '24 Hour Urine Volume (263)',
 'Hemoglobin',
 'Total Leucocytic Count',
 'Hematocrit',
 'MCV',
 'Glucose(Urine Physical Examination)',
 'Urea in Serum',
 'Prostatic Specific Antigen (PSA) Total',
 'Testosterone (Total)',
 'Alkaline Phosphatase',
 'Total Protein in Serum',
 'Estimated Glomerular Filtration Rate(eGFR)',
 'Anti CCP Abs',
 ' BUN/Creatinine Ratio',
 'Ketones',
 'MCHC',
 'pH(Urine Physical Examination)',
 'Amorphous Elements',
 'Blood and Haemoglobin',
 'Epithelial Cells / HPF',
 'Casts(Urine Microscopic Examination :)',
 'Bilirubin',
 'Chloride in Serum',
 'Cholesterol',
 'T. Cholesterol/HDL',
 'Urobilinogen',
 'R.B.Cs / HPF',
```

```
df[ 'HEIGHT'].isnull().sum().sum()
```

```
np.int64(0)
```

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

```
array([168, 156, 154, 167, 161, 162, 160, 157, 148, 159, 158, 164, 171,
       170, 189, 172, 151, 165, 150, 163, 127, 169, 153, 142, 144, 128,
       174, 146, 173, 155, 181, 176, 179, 180, 182, 175, 122,  86,  80,
       149, 178, 145, 183, 140, 166, 152, 177, 184, 186, 138, 147, 188,
       143, 141,  75, 132, 139,  69, 133, 113, 137, 136, 125, 115, 118,
       117, 185, 109, 114,  51, 121,  65,  63, 194,  50,  95, 192,  89,
       134, 108, 103, 187,  66,  84, 100,  78, 101, 110, 130, 102,  71,
        94, 190, 195,  88,  53,  82, 105, 116, 123, 120, 111, 131, 119,
       106,  76, 135, 107,  85,  61,  60,  97,  93, 124,  70, 193,  52,
       197,  98,  87, 112, 196,  57, 129,  72,  67,  83,  73,  62, 104,
       191,  77,  79,  96,  74,  59, 126,  58,  68,  55, 198,  90,  56,
       199,  81,  91,  64,  99,  54,  92, 200, 203, 202])
```

```
df[ 'WEIGHT'].isnull().sum().sum()
```

```
np.int64(0)
```

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

```
array([ 84.12,  84.09,  75.3 , ...,  93.58, 118.18,  67.13])
```

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

```
array([29.8 , 29.79, 30.94, ..., 43.35, 33.85, 45.5 ])
```

```
df[ 'BMI'].isnull().sum().sum()
```

```
np.int64(0)
```

```
df[ 'HEIGHT'].isnull().sum().sum()
```

```
np.int64(0)
```

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

```
array([168, 156, 154, 167, 161, 162, 160, 157, 148, 159, 158, 164, 171,
       170, 189, 172, 151, 165, 150, 163, 127, 169, 153, 142, 144, 128,
       174, 146, 173, 155, 181, 176, 179, 180, 182, 175, 122,  86,  80,
       149, 178, 145, 183, 140, 166, 152, 177, 184, 186, 138, 147, 188,
       143, 141,  75, 132, 139,  69, 133, 113, 137, 136, 125, 115, 118,
       117, 185, 109, 114,  51, 121,  65,  63, 194,  50,  95, 192,  89,
       134, 108, 103, 187,  66,  84, 100,  78, 101, 110, 130, 102,  71,
        94, 190, 195,  88,  53,  82, 105, 116, 123, 120, 111, 131, 119,
       106,  76, 135, 107,  85,  61,  60,  97,  93, 124,  70, 193,  52,
       197,  98,  87, 112, 196,  57, 129,  72,  67,  83,  73,  62, 104,
       191,  77,  79,  96,  74,  59, 126,  58,  68,  55, 198,  90,  56,
       199,  81,  91,  64,  99,  54,  92, 200, 203, 202])
```

```
Start coding or generate with AI.
```