heartandairquality

April 29, 2025

[160]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

a. Data cleaning

[162]:
```
air_df = pd.read_csv("AirQuality.csv")
air_df.head()
```

[162]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	Wind	PT08.S2(NMHC)	\
0	10-03-2004	18:00:00	2.6	1360	150	11.9	1046	
1	10-03-2004	19:00:00	2.0	1292	112	9.4	955	
2	10-03-2004	20:00:00	2.2	1402	88	9.0	939	
3	10-03-2004	21:00:00	2.2	1376	80	9.2	948	
4	10-03-2004	22:00:00	1.6	1272	51	6.5	836	

	Solor.R	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	Ozone	Temp	RH	AH
0	166	1056	113	1692	1268	13.6	48.9	0.7578
1	103	1174	92	1559	972	13.3	47.7	0.7255
2	131	1140	114	1555	1074	11.9	54.0	0.7502
3	172	1092	122	1584	1203	11.0	60.0	0.7867
4	131	1205	116	1490	1110	11.2	59.6	0.7888

[163]:
```
air_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           9357 non-null   object
 1   Time           9357 non-null   object
 2   CO(GT)         9357 non-null   float64
 3   PT08.S1(CO)    9357 non-null   int64
 4   NMHC(GT)       9357 non-null   int64
```

```
 5   Wind           9357 non-null   float64
 6   PT08.S2(NMHC)  9357 non-null   int64
 7   Solor.R        9357 non-null   int64
 8   PT08.S3(NOx)   9357 non-null   int64
 9   NO2(GT)        9357 non-null   int64
 10  PT08.S4(NO2)   9357 non-null   int64
 11  Ozone          9357 non-null   int64
 12  Temp           9357 non-null   float64
 13  RH             9357 non-null   float64
 14  AH             9357 non-null   float64
dtypes: float64(5), int64(8), object(2)
memory usage: 1.1+ MB
```

[164]:
```
air_df.shape
```

[164]:
```
(9357, 15)
```

[165]:
```
heart_df = pd.read_csv("heartdisease.csv")
heart_df.head()
```

[165]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0

[166]:
```
heart_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
```

```
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    object
 12  thal      303 non-null    object
 13  num       303 non-null    int64
dtypes: float64(1), int64(11), object(2)
memory usage: 33.3+ KB
```

[167]:
```
heart_df.shape
```

[167]:
```
(303, 14)
```

[168]:
```
# Removing null values
air_df.dropna(inplace=True)
heart_df.dropna(inplace=True)
```

[169]:
```
# Removing duplicates
air_df.drop_duplicates(inplace=True)
heart_df.drop_duplicates(inplace=True)
```

[170]:
```
air_df.shape
```

[170]:
```
(9357, 15)
```

[171]:
```
heart_df.shape
```

[171]:
```
(303, 14)
```

[172]:
```
# Checking for missing values
air_df.isnull().sum()
```

[172]:
```
Date             0
Time             0
CO(GT)           0
PT08.S1(CO)      0
NMHC(GT)         0
Wind             0
PT08.S2(NMHC)    0
Solor.R          0
PT08.S3(NOx)     0
NO2(GT)          0
PT08.S4(NO2)     0
Ozone            0
Temp             0
RH               0
```

```
AH              0
dtype: int64
```

[174]:
```
heart_df.isnull().sum()
```

[174]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
num         0
dtype: int64
```

b. Data Integration

[176]:
```
# Create artificial IDs for integration purpose
air_df['ID'] = range(1, len(air_df) + 1)
heart_df['ID'] = range(1, len(heart_df) + 1)

# Merge on the ID column (inner join for same size, or left/right based on␣
 ↪use-case)
integrated_df = pd.merge(air_df, heart_df, on='ID', how='inner')

# Checking the result
print("Integrated Dataset Shape:", integrated_df.shape)
integrated_df.head()
```

```
Integrated Dataset Shape: (303, 30)
```

[176]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	Wind	PT08.S2(NMHC)	\
0	10-03-2004	18:00:00	2.6	1360	150	11.9	1046	
1	10-03-2004	19:00:00	2.0	1292	112	9.4	955	
2	10-03-2004	20:00:00	2.2	1402	88	9.0	939	
3	10-03-2004	21:00:00	2.2	1376	80	9.2	948	
4	10-03-2004	22:00:00	1.6	1272	51	6.5	836	

	Solor.R	PT08.S3(NOx)	NO2(GT)	…	chol	fbs	restecg	thalach	exang	\
0	166	1056	113	…	233	1	2	150	0	
1	103	1174	92	…	286	0	2	108	1	

```
2      131          1140      114  …   229    0        2      129      1
3      172          1092      122  …   250    0        0      187      0
4      131          1205      116  …   204    0        2      172      0

   oldpeak  slope  ca  thal  num
0      2.3      3   0     6    0
1      1.5      2   3     3    2
2      2.6      2   2     7    1
3      3.5      3   0     3    0
4      1.4      1   0     3    0

[5 rows x 30 columns]
```

c. Data Transformation

[178]:
```
# Check if any columns are still non-numeric
non_numeric_cols = integrated_df.select_dtypes(exclude=['number']).columns
print("Non-numeric columns:", non_numeric_cols.tolist())
```

```
Non-numeric columns: ['Date', 'Time', 'ca', 'thal']
```

[179]:
```
# Step 1: Drop Date and Time
if 'Date' in integrated_df.columns:
    integrated_df.drop(columns=['Date'], inplace=True)
if 'Time' in integrated_df.columns:
    integrated_df.drop(columns=['Time'], inplace=True)
```

[180]:
```
# Step 2: Encode 'ca' and 'thal'
label_enc = LabelEncoder()
for col in ['ca', 'thal']:
    if col in integrated_df.columns:
        integrated_df[col] = label_enc.fit_transform(integrated_df[col].
 ↪astype(str))
```

[181]:
```
integrated_df.head()
```

[181]:
```
   CO(GT)  PT08.S1(CO)  NMHC(GT)  Wind  PT08.S2(NMHC)  Solor.R  PT08.S3(NOx)  \
0     2.6         1360       150  11.9           1046      166          1056
1     2.0         1292       112   9.4            955      103          1174
2     2.2         1402        88   9.0            939      131          1140
3     2.2         1376        80   9.2            948      172          1092
4     1.6         1272        51   6.5            836      131          1205

   NO2(GT)  PT08.S4(NO2)  Ozone  …  chol  fbs  restecg  thalach  exang  \
0      113          1692   1268  …   233    1        2      150      0
1       92          1559    972  …   286    0        2      108      1
2      114          1555   1074  …   229    0        2      129      1
3      122          1584   1203  …   250    0        0      187      0
```

```
4      116          1490   1110  …   204    0        2      172      0

   oldpeak  slope  ca  thal  num
0      2.3      3   0     1    0
1      1.5      2   3     0    2
2      2.6      2   2     2    1
3      3.5      3   0     0    0
4      1.4      1   0     0    0

[5 rows x 28 columns]
```

[182]:
```
# Step 4: Re-scale
scaler = StandardScaler()
transformed_array = scaler.fit_transform(integrated_df)
transformed_df = pd.DataFrame(transformed_array, columns=integrated_df.columns)

transformed_df.head()
```

[182]:
```
      CO(GT)  PT08.S1(CO)  NMHC(GT)      Wind  PT08.S2(NMHC)   Solor.R  \
0  0.220853     0.243725  0.522726  0.083674       0.234046  0.183713
1  0.206751    -0.027733  0.341681 -0.237904      -0.097014 -0.356390
2  0.211452     0.411390  0.227336 -0.289357      -0.155222 -0.116345
3  0.211452     0.307597  0.189222 -0.263631      -0.122480  0.235151
4  0.197350    -0.107574  0.051056 -0.610936      -0.529938 -0.116345

   PT08.S3(NOx)   NO2(GT)  PT08.S4(NO2)     Ozone  …      chol       fbs  \
0      0.172837  0.249821      0.140058  0.331721  … -0.264900  2.394438
1      0.594768 -0.036029     -0.327138 -0.414526  …  0.760415 -0.417635
2      0.473194  0.263432     -0.341189 -0.157373  … -0.342283 -0.417635
3      0.301561  0.372327     -0.239319  0.167849  …  0.063974 -0.417635
4      0.705614  0.290656     -0.569517 -0.066614  … -0.825922 -0.417635

    restecg   thalach     exang   oldpeak     slope        ca      thal  \
0  1.016684  0.017197 -0.696631  1.087338  2.274579 -0.713129  0.153317
1  1.016684 -1.821905  1.435481  0.397182  0.649113  2.274127 -0.879017
2  1.016684 -0.902354  1.435481  1.346147  0.649113  1.278375  1.185650
3 -0.996749  1.637359 -0.696631  2.122573  2.274579 -0.713129 -0.879017
4  1.016684  0.980537 -0.696631  0.310912 -0.976352 -0.713129 -0.879017

        num
0 -0.764198
1  0.866450
2  0.051126
3 -0.764198
4 -0.764198

[5 rows x 28 columns]
```

[183]:
```
transformed_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 28 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   CO(GT)          303 non-null    float64
 1   PT08.S1(CO)     303 non-null    float64
 2   NMHC(GT)        303 non-null    float64
 3   Wind            303 non-null    float64
 4   PT08.S2(NMHC)   303 non-null    float64
 5   Solor.R         303 non-null    float64
 6   PT08.S3(NOx)    303 non-null    float64
 7   NO2(GT)         303 non-null    float64
 8   PT08.S4(NO2)    303 non-null    float64
 9   Ozone           303 non-null    float64
 10  Temp            303 non-null    float64
 11  RH              303 non-null    float64
 12  AH              303 non-null    float64
 13  ID              303 non-null    float64
 14  age             303 non-null    float64
 15  sex             303 non-null    float64
 16  cp              303 non-null    float64
 17  trestbps        303 non-null    float64
 18  chol            303 non-null    float64
 19  fbs             303 non-null    float64
 20  restecg         303 non-null    float64
 21  thalach         303 non-null    float64
 22  exang           303 non-null    float64
 23  oldpeak         303 non-null    float64
 24  slope           303 non-null    float64
 25  ca              303 non-null    float64
 26  thal            303 non-null    float64
 27  num             303 non-null    float64
dtypes: float64(28)
memory usage: 66.4 KB
```

d. Error Correcting

[185]:
```
# Step 1: Check for NaN or Infinite values
print("NaNs in dataset:", transformed_df.isnull().sum().sum())
print("Infinite values:", np.isinf(transformed_df).sum().sum())
```

```
NaNs in dataset: 0
Infinite values: 0
```

[186]:
```
# Step 2: Z-score outlier detection
z_scores = np.abs((transformed_df - transformed_df.mean()) / transformed_df.
 ↪std())
outliers = (z_scores > 3).sum().sum()
print("Total potential outlier values:", outliers)
```

```
Total potential outlier values: 57
```

[187]:
```
transformed_df = transformed_df.clip(lower=-3, upper=3)
```

[188]:
```
# Reset index after corrections
transformed_df.reset_index(drop=True, inplace=True)

print("Error correction completed. Data is clean and model-ready.")
transformed_df.head()
```

```
Error correction completed. Data is clean and model-ready.
```

[188]:

	CO(GT)	PT08.S1(CO)	NMHC(GT)	Wind	PT08.S2(NMHC)	Solor.R	\
0	0.220853	0.243725	0.522726	0.083674	0.234046	0.183713	
1	0.206751	-0.027733	0.341681	-0.237904	-0.097014	-0.356390	
2	0.211452	0.411390	0.227336	-0.289357	-0.155222	-0.116345	
3	0.211452	0.307597	0.189222	-0.263631	-0.122480	0.235151	
4	0.197350	-0.107574	0.051056	-0.610936	-0.529938	-0.116345	

	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	Ozone	…	chol	fbs	\
0	0.172837	0.249821	0.140058	0.331721	…	-0.264900	2.394438	
1	0.594768	-0.036029	-0.327138	-0.414526	…	0.760415	-0.417635	
2	0.473194	0.263432	-0.341189	-0.157373	…	-0.342283	-0.417635	
3	0.301561	0.372327	-0.239319	0.167849	…	0.063974	-0.417635	
4	0.705614	0.290656	-0.569517	-0.066614	…	-0.825922	-0.417635	

	restecg	thalach	exang	oldpeak	slope	ca	thal	\
0	1.016684	0.017197	-0.696631	1.087338	2.274579	-0.713129	0.153317	
1	1.016684	-1.821905	1.435481	0.397182	0.649113	2.274127	-0.879017	
2	1.016684	-0.902354	1.435481	1.346147	0.649113	1.278375	1.185650	
3	-0.996749	1.637359	-0.696631	2.122573	2.274579	-0.713129	-0.879017	
4	1.016684	0.980537	-0.696631	0.310912	-0.976352	-0.713129	-0.879017	

	num
0	-0.764198
1	0.866450
2	0.051126
3	-0.764198
4	-0.764198

[5 rows x 28 columns]

[189]:
```
transformed_df.shape
```

[189]:
```
(303, 28)
```

e. Data Model Building

[191]:
```
# Step 1: Features and Target
X = transformed_df.iloc[:, :-1]
y = transformed_df.iloc[:, -1]
```

[192]:
```
# Step 2: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[193]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[193]:
```
((242, 27), (61, 27), (242,), (61,))
```

[194]:
```
# Step 3: Initialize and Train Regressor
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

[194]:
```
RandomForestRegressor(random_state=42)
```

[195]:
```
# Step 4: Predict
y_pred = model.predict(X_test)
```

[196]:
```
# Step 5: Evaluation
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
```

```
C:\Users\amans\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:483:
FutureWarning: 'squared' is deprecated in version 1.4 and will be removed in
1.6. To calculate the root mean squared error, use the
function'root_mean_squared_error'.
  warnings.warn(
```

[197]:
```
print("  Model Evaluation (Regression):")
print(f"RMSE: {rmse:.4f}")
print(f"R² Score: {r2:.4f}")
```

```
 Model Evaluation (Regression):
RMSE: 0.7122
R² Score: 0.5411
```

[ ]:

[ ]: