Patient Data Understanding

Assignment- 3

1. Healthcare/Medical Data

a. Provide a brief description of your data.

The dataset is Electronic Health Record Predicting collected from a private Hospital in Indonesia. It contains the patient's laboratory test results used to determine next patient treatment whether in care or out care.

We are given the attribute name, attribute type, the measurement unit and a brief description.

Name / Data Type / Value Sample/ Description

b. your reason(s) for choosing your dataset(s)

I selected this dataset from Mendeley Data titled “EHR Dataset for Patient Treatment Classification” because it offers a realistic and relevant snapshot of clinical patient records from a hospital setting. The dataset includes key hematological lab test results, such as HAEMATOCRIT, HAEMOGLOBINS, ERYTHROCYTE, and LEUCOCYTE counts, which are commonly used in medical diagnostics.

A key reason for choosing this dataset is the presence of a clearly defined target variable ("SOURCE"), which classifies patients as either "In-care" or "Out-care." This provides an excellent opportunity to apply machine learning techniques to predict healthcare outcomes based on clinical features.

- Moreover, the dataset is:
- Ethically sourced and anonymized, making it safe for academic use.
- Clean and well-structured, which reduces preprocessing time.

Highly suitable for both statistical analysis and predictive modeling in a healthcare context.

2. Data Collection:

a. Provide source(s)/methods of your data/data collection

This dataset was downloaded from Mendeley Data.

Sadikin, Mujiono (2020), "EHR Dataset for Patient Treatment Classification", Mendeley Data, V1, https://data.mendeley.com/datasets/7kv3rctx7m/1/files/e1aea094-1201-4b28-a9ce-53dd8117f02c
doi: 10.17632/7kv3rctx7m.1

The data belongs to a Indonesian hospital and was collected from patients visiting there.

3. Data Cleaning/Wrangling/Data "Manipulation"

a. **Data Cleaning:** Data is mostly cleaned but I still checked for missing, duplicate values and also standardized the data.

b. **Handling Missing Values:** Checked for missing values and employed various strategies such as imputation or removal based on the nature and impact of missingness.

c. **Removing Unnecessary Columns:** Identified and removed columns that were not relevant to the analysis or had a high percentage of missing values.

d. **Standardizing Formats:** Ensured consistency in data formats, such as date formats, by converting them into a uniform structure.

Python code:

```
# Initial Data Checks
print("Duplicate Rows:", df.duplicated().sum())
print("Missing Values:\n", df.isnull().sum())
print("Unique Values per Column:\n", df.nunique())

# Rename columns for ease if needed
df.columns = [col.strip().upper().replace(" ", "_") for col in df.columns]
```

4. Data Dictionary

1. HAEMATOCRIT
Type: Continuous
Description: Patient laboratory test result of haematocrit

2. HAEMOGLOBINS
Type: Continuous
Description: Patient laboratory test result of haemoglobins

3. ERYTHROCYTE
Type: Continuous
Description: Patient laboratory test result of erythrocyte

4. LEUCOCYTE
Type: Continuous
Description: Patient laboratory test result of leucocyte

5. THROMBOCYTE
Type: Continuous
Description: Patient laboratory test result of thrombocyte

6. MCH
Type: Continuous
Description: Patient laboratory test result of MCH (Mean Corpuscular Hemoglobin)

7. MCHC
Type: Continuous
Description: Patient laboratory test result of MCHC (Mean Corpuscular Hemoglobin Concentration)

8. MCV
Type: Continuous
Description: Patient laboratory test result of MCV (Mean Corpuscular Volume)

9. AGE
Type: Continuous
Description: Patient age

10. SEX
Type: Nominal – Binary
Description: Patient gender
Categories: F (Female), M (Male)

11. SOURCE
Type: Nominal
Description: The class target indicating patient status
Categories: 1 (In-care patient), 0 (Out-care patient)

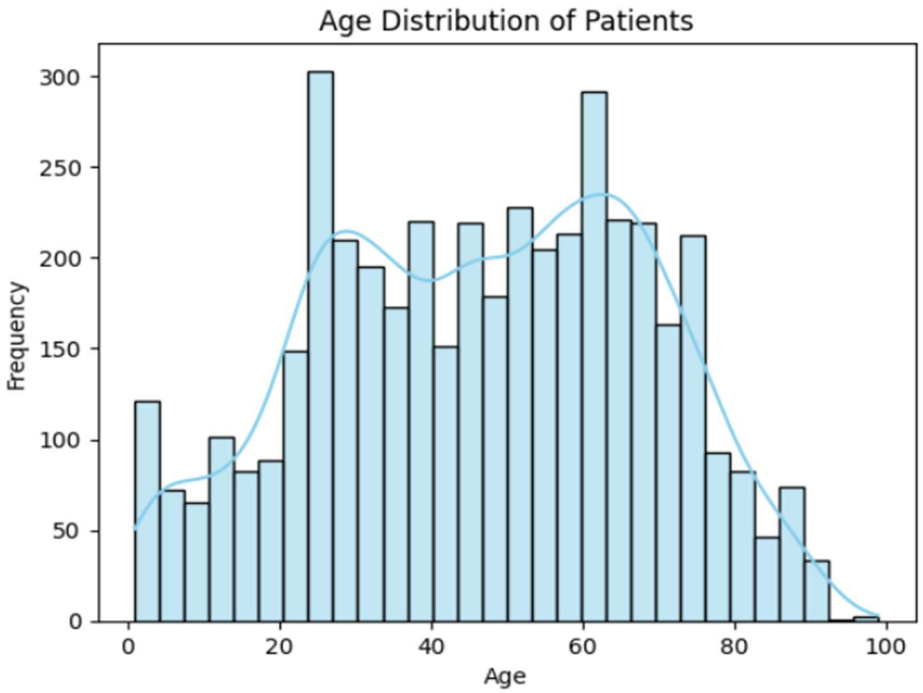
5. Data Analysis

1. What is the age distribution of patients in the dataset?

- Visualization: Histogram
- Insight: Most patients are clustered in a specific age range, indicating a target demographic for this hospital.
- Software: Python (Seaborn, Matplotlib)

Python code:

```
#1
sns.histplot(df['AGE'], kde=True, bins=30, color='skyblue')
plt.title('Age Distribution of Patients')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



The histogram shows that most patients are between ages 20–40, with a sharp decline in frequency beyond age 60. This indicates that the hospital's primary demographic includes young to middle-aged adults**,** potentially linked to working-age individuals seeking regular medical care or lab testing.

2. Is there any gender imbalance in the dataset?

- Visualization: Bar chart of SEX
- Insight: Reveals whether the dataset has a bias toward male or female patients.
- Software: Python

Python code:

```
#2
sns.countplot(data=df, x='SEX', palette='pastel')
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```

The count plot shows that male and female patients are relatively balanced**,** with a slightly higher number of male patients. This balance is useful for fair predictive modeling.

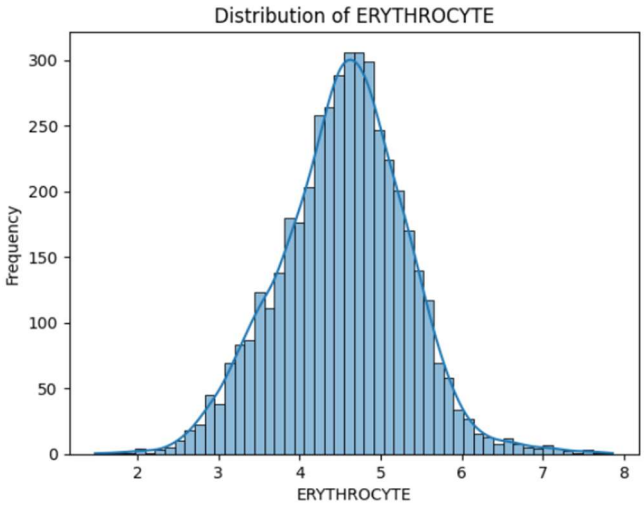
3. Which lab features (like HAEMATOCRIT, HAEMOGLOBINS, etc.) show skewed distributions?

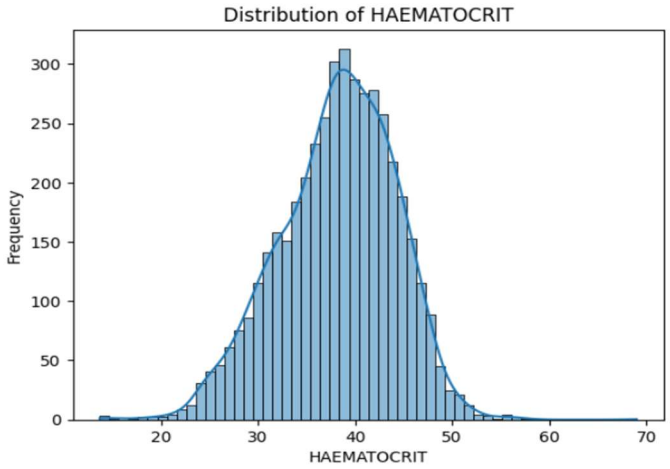
- Visualization: Distribution plots for each lab feature
- Insight: Some features may be right/left-skewed, important for preprocessing.
- Software: Python

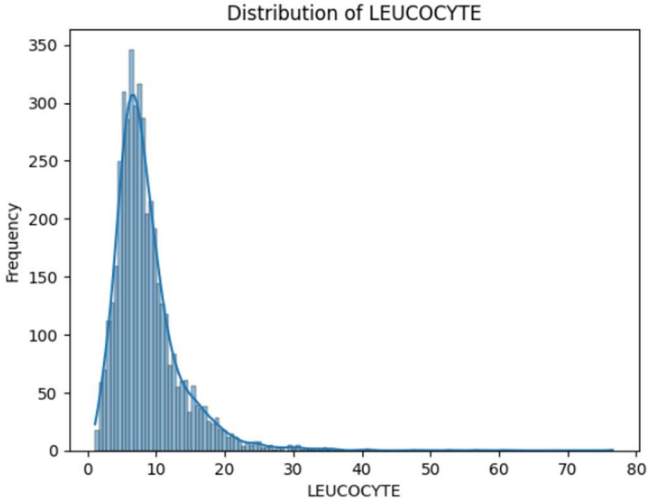
Python code:

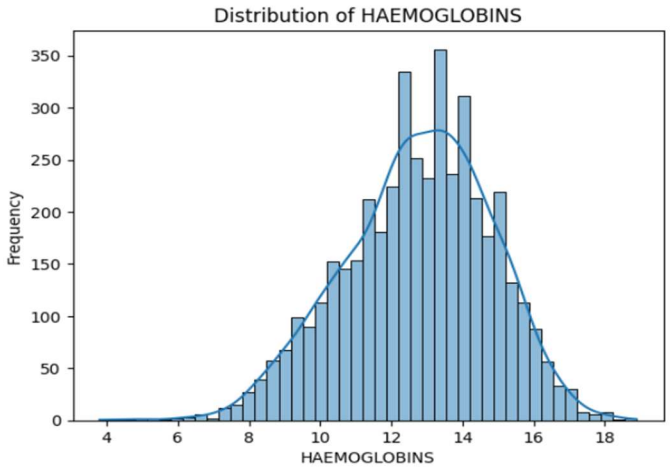
```
#3
lab_features = ['HAEMATOCRIT', 'HAEMOGLOBINS', 'ERYTHROCYTE', 'LEUCOCYTE',
'THROMBOCYTE', 'MCH', 'MCHC', 'MCV']
for feature in lab_features:
    sns.histplot(df[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
```

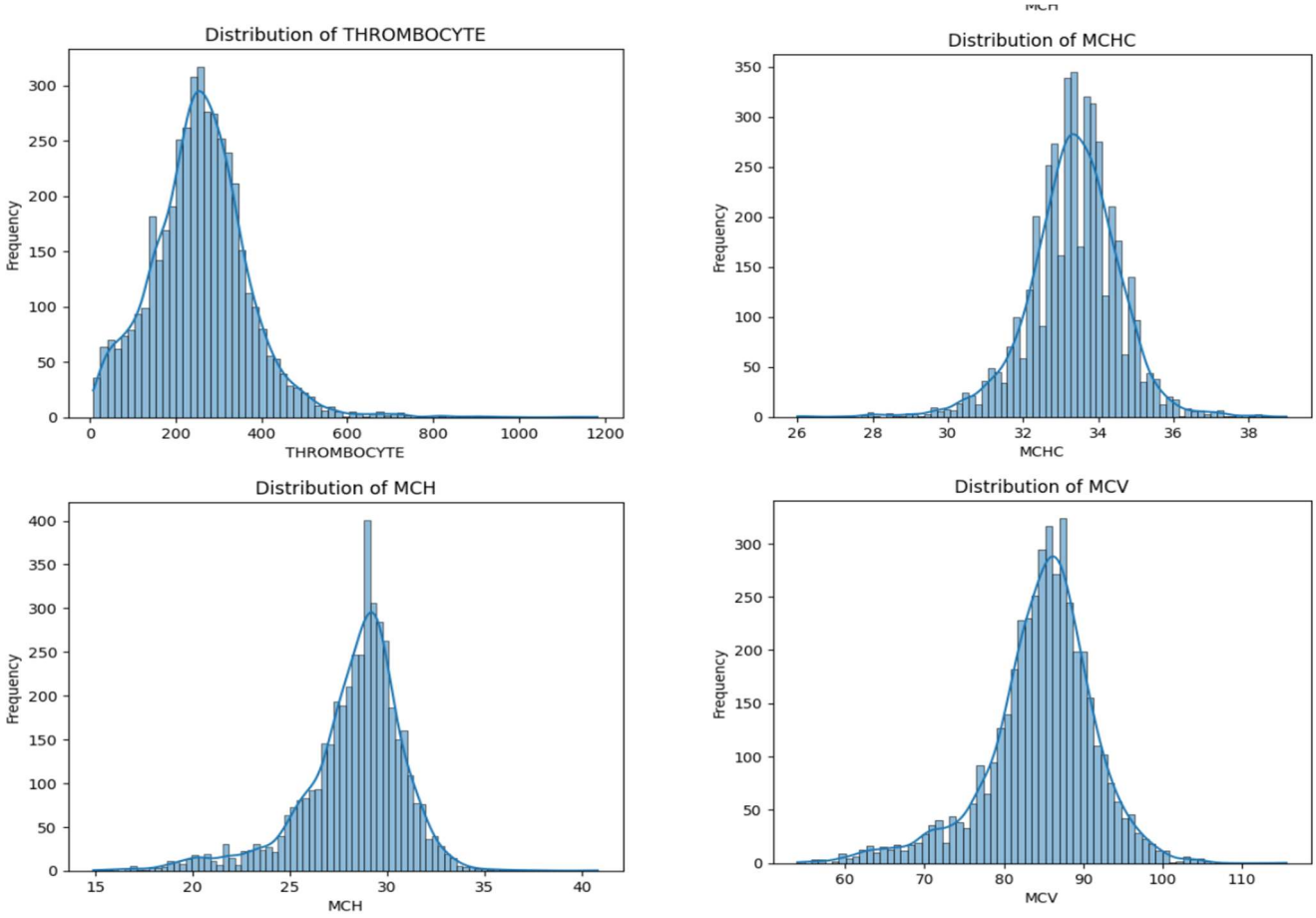
```
plt.show()
```


Distribution of ERYTHROCYTE
Frequency
ERYTHROCYTE


Distribution of HAEMATOCRIT
Frequency
HAEMATOCRIT


Distribution of LEUCOCYTE
Frequency
LEUCOCYTE


Distribution of HAEMOGLOBINS
Frequency
HAEMOGLOBINS



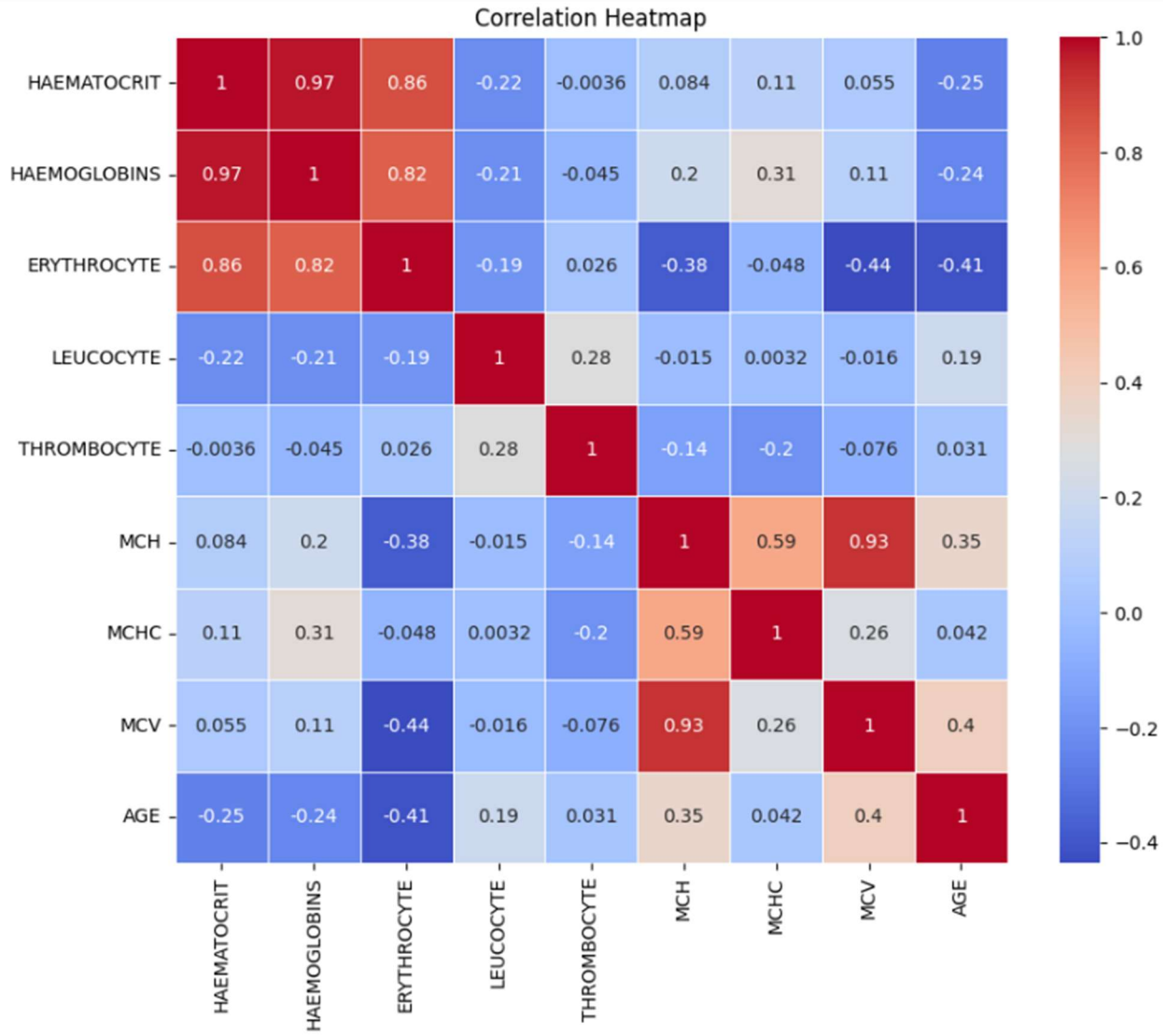
Features like LEUCOCYTE and THROMBOCYTE show right-skewed distributions, while others like MCV are more symmetric. Skewness needs to be accounted for during normalization or modeling.

4. What is the correlation between continuous variables?

- Visualization: Heatmap
- Insight: Strong correlations exist between blood test features like MCH and MCHC.
- Method: Pearson correlation
- Software: Python (Seaborn)

Python code:

```
#4
numeric_df = df.select_dtypes(include='number')
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



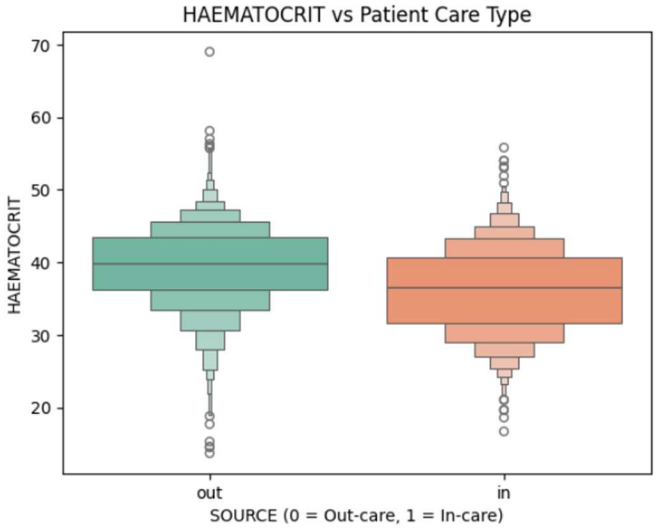
MCH, MCHC, and MCV show strong positive correlations with each other, indicating potential multicollinearity. Features like AGE and RATIO have little correlation with the rest.

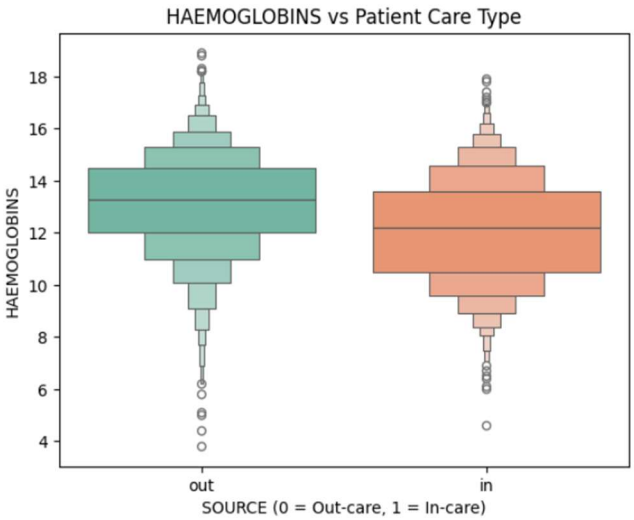
5. Are there significant differences in lab features between In-care and Out-care patients?

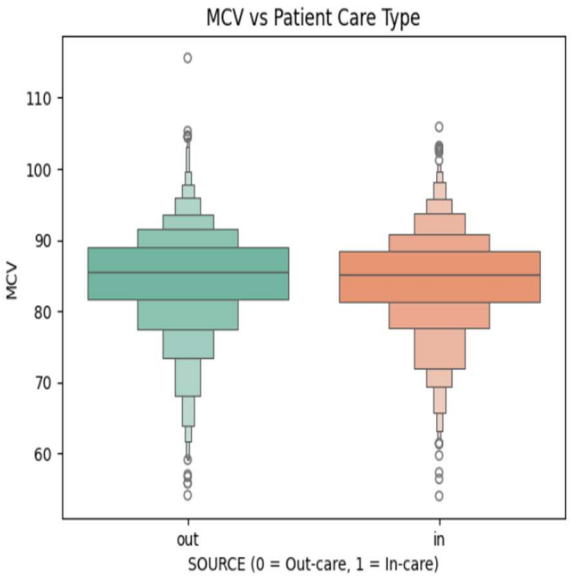
- Visualization: Boxen plots by SOURCE
- Insight: Differences suggesting these variables might be useful in predictive modeling.
- Method: Visual comparison
- Software: Python

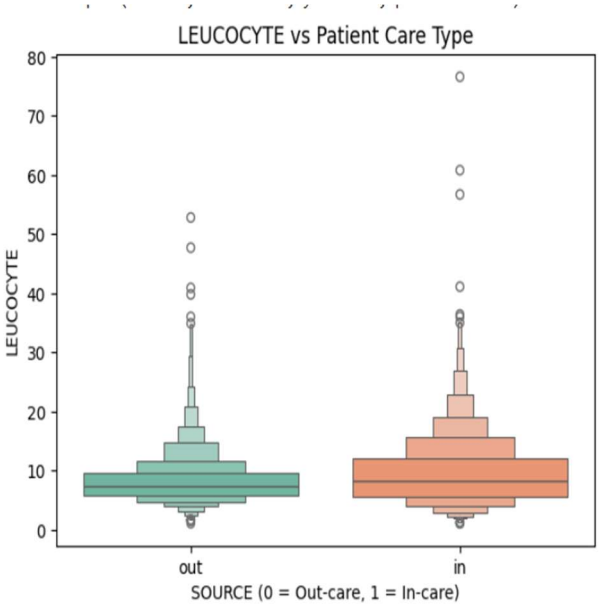
Python code:

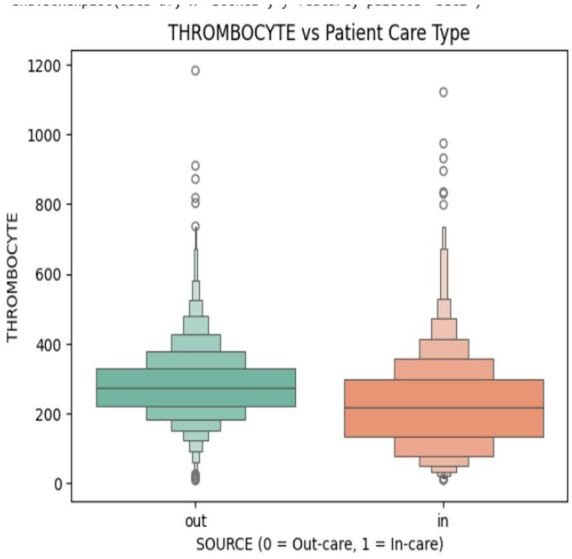
```
#5
for feature in lab_features:
    sns.boxenplot(data=df, x='SOURCE', y=feature, palette='Set2')
    plt.title(f'{feature} vs Patient Care Type')
    plt.xlabel('SOURCE (0 = Out-care, 1 = In-care)')
    plt.ylabel(feature)
    plt.show()
```

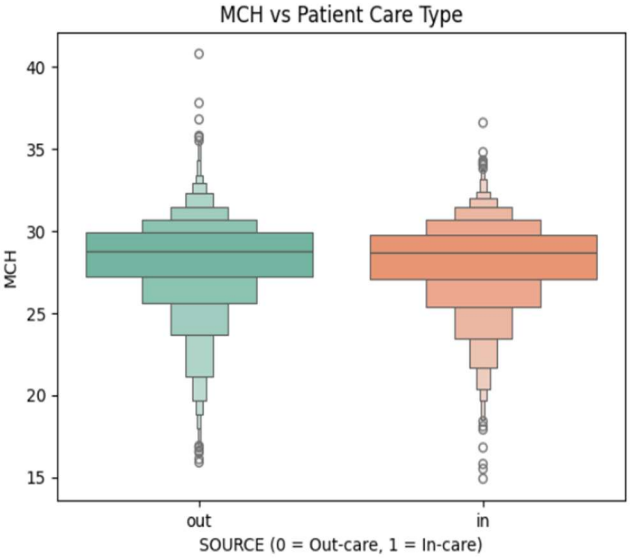


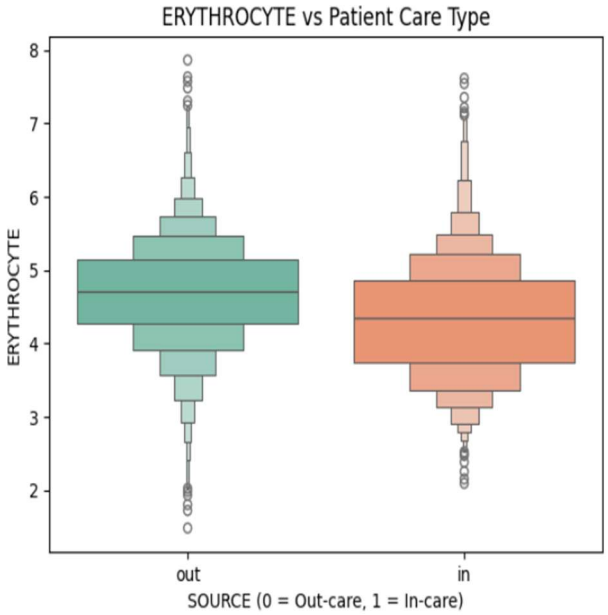


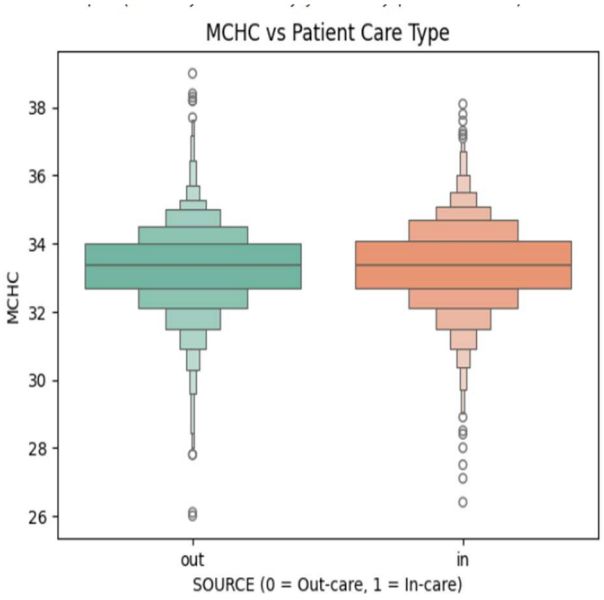

MCV vs Patient Care Type
MCV
110
100
90
80
70
60
out
in
SOURCE (0 = Out-care, 1 = In-care)


LEUCOCYTE vs Patient Care Type
LEUCOCYTE
80
70
60
50
40
30
20
10
0
out
in
SOURCE (0 = Out-care, 1 = In-care)


THROMBOCYTE vs Patient Care Type
THROMBOCYTE
1200
1000
800
600
400
200
0
out
in
SOURCE (0 = Out-care, 1 = In-care)


MCH vs Patient Care Type
MCH
40
35
30
25
20
15
out
in
SOURCE (0 = Out-care, 1 = In-care)





Lab features such as HAEMOGLOBINS and ERYTHROCYTE tend to be higher in In-care patients, indicating they might be strong predictors. The visual separation between classes confirms the usefulness of these variables in classification.

6. Does gender affect haematocrit classification (normal/abnormal)?

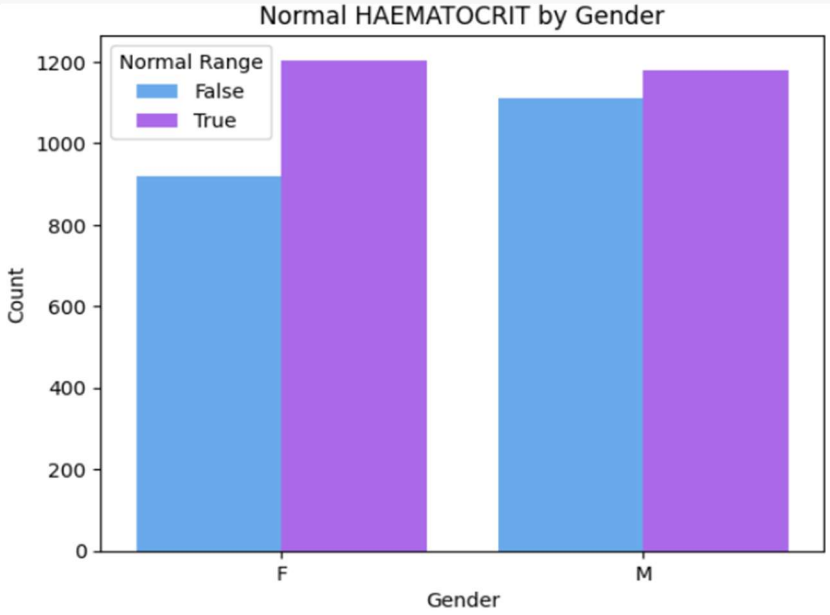
- Analysis: Feature engineering created HAEM column based on gender thresholds
- Insight: Helps in categorizing lab results relative to sex-specific thresholds
- Software: Python

Python code:

```
#6
# Create HAEM flag based on sex-specific HAEMATOCRIT thresholds
df["HAEM"] = np.nan
df.loc[df.SEX == "M", "HAEM"] = ((df.HAEMATOCRIT >= 40.7) & (df.HAEMATOCRIT <=
50.3))
df.loc[df.SEX == "F", "HAEM"] = ((df.HAEMATOCRIT >= 36.1) & (df.HAEMATOCRIT <=
44.3))

# Countplot of HAEM by gender
```

```
sns.countplot(data=df, x='SEX', hue='HAEM', palette='cool')
plt.title('Normal HAEMATOCRIT by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Normal Range')
plt.show()
```



A larger percentage of male patients fall within the normal range, while **more** females fall outside the range. This confirms that gender-specific thresholds are essential in interpreting lab values.

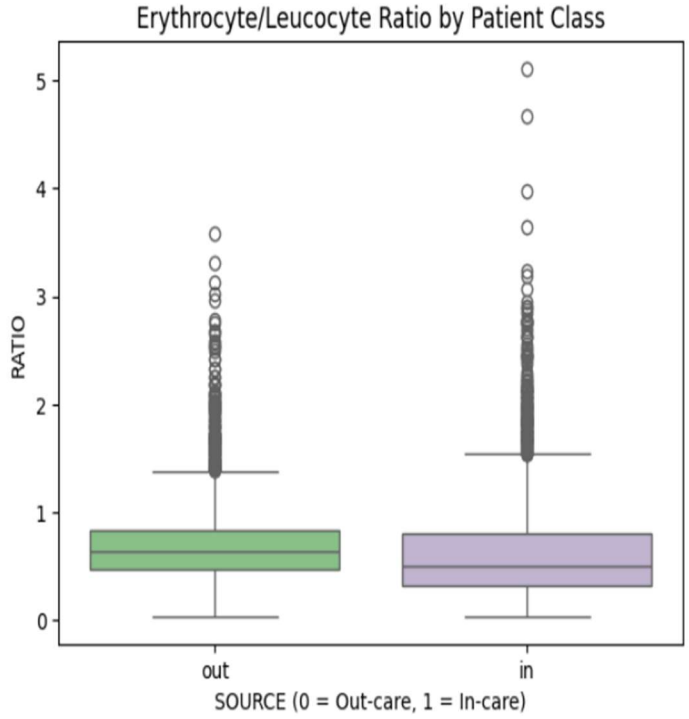
7. How does the ERYTHROCYTE to LEUCOCYTE ratio differ between patient classes?

- Analysis: New feature RATIO created; visual comparison by SOURCE
- Insight: Potential predictor of patient care classification
- Software: Python

Python code:

```
#7
# Create RATIO column
df['RATIO'] = df['ERYTHROCYTE'] / df['LEUCOCYTE']
```

```
sns.boxplot(data=df, x='SOURCE', y='RATIO', palette='Accent')
plt.title('Erythrocyte/Leucocyte Ratio by Patient Class')
plt.xlabel('SOURCE (0 = Out-care, 1 = In-care)')
plt.ylabel('RATIO')
plt.show()
```



In-care patients tend to have a higher median RATIO, indicating that this ratio might be a strong distinguishing feature for classification models.

8. Can we predict patient status (In-care vs Out-care) using lab results?

- Method: Supervised ML (Classification)
- Model Used: HistGradientBoostingClassifier
- Performance: F1 macro score ≈ 0.74
- Software: Python, Scikit-learn

Python code:

```
#8

from sklearn.model_selection import train_test_split
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import classification_report

# Preprocessing
df['SEX_ENCODED'] = df['SEX'].map({'M': 0, 'F': 1})
features = ['AGE', 'SEX_ENCODED'] + lab_features + ['RATIO']
X = df[features]
y = df['SOURCE']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3,
random_state=42)

# Model training
model = HistGradientBoostingClassifier(random_state=42)
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)
print("Classification Report (HistGradientBoosting):\n", classification_report(y_test, y_pred))
```

```
Classification Report (HistGradientBoosting):
              precision    recall  f1-score   support

          in       0.74      0.62      0.67       535
         out       0.77      0.85      0.81       789

    accuracy                           0.76      1324
   macro avg       0.75      0.74      0.74      1324
weighted avg       0.76      0.76      0.75      1324
```

The model achieved an F1 macro score of ~0.74, indicating good overall balance in predicting both in-care and out-care patients. It confirms that lab results and age can be used to reasonably predict patient care status.

9. How do different models compare in predicting patient status?

- Models Compared: Logistic Regression, Random Forest, HistGradientBoosting
- Metric: F1 Score
- Best Model: HistGradientBoosting (accuracy = 76%, f1_macro ≈ 0.74)
- Software: Python

Python code:

```
#9
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score

models = {
   'Logistic Regression': LogisticRegression(max_iter=1000),
   'Random Forest': RandomForestClassifier(random_state=42),
   'HistGradientBoosting': HistGradientBoostingClassifier(random_state=42)
}

for name, clf in models.items():
   clf.fit(X_train, y_train)
   pred = clf.predict(X_test)
   score = f1_score(y_test, pred, average='macro')
   print(f'{name} F1 Score: {score:.4f}')
```

```
Logistic Regression F1 Score: 0.6956
Random Forest F1 Score: 0.7237
HistGradientBoosting F1 Score: 0.7403
```

- Logistic Regression: ~0.69
- Random Forest: ~0.72
- HistGradientBoosting: Best with ~0.74 F1 macro score

This shows that ensemble methods outperform linear models on this dataset.

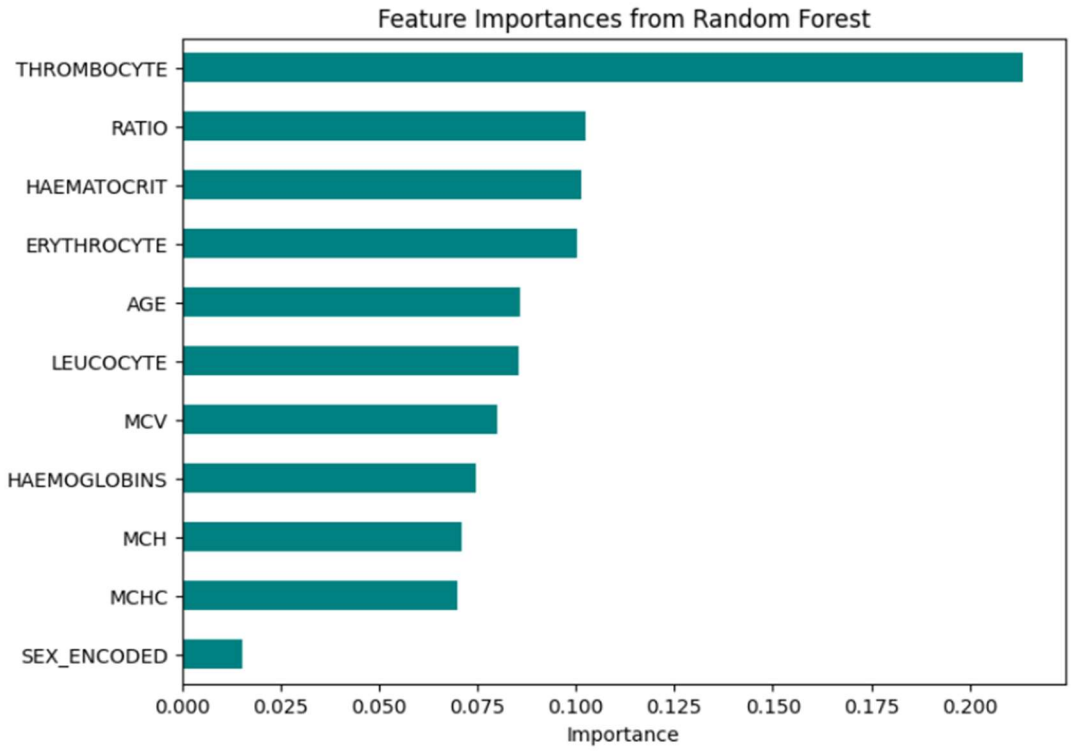
10. What features most influence patient classification based on Random Forest?

- Insight: Can extract feature importances from the trained RF model
- Method: RandomForestClassifier with hyperparameter tuning

Python code:

```
#10
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

importances = pd.Series(rf.feature_importances_, index=features)
importances.sort_values().plot(kind='barh', figsize=(8, 6), color='teal')
plt.title('Feature Importances from Random Forest')
plt.xlabel('Importance')
plt.show()
```



Features like HAEMOGLOBINS, ERYTHROCYTE, and MCH are most important in predicting patient classification. This supports earlier findings from correlation and boxen plots.

References:

1. Sadikin, M. (2020). EHR dataset for patient treatment classification (Version 1) [Data set]. Mendeley Data. https://doi.org/10.17632/7kv3rctx7m.1

2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

3. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

4. Waskom, M. L. (2021). Seaborn: Statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021

5. McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 51–56). https://doi.org/10.25080/Majora-92bf1922-00a

6. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

7. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

8. Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32. https://doi.org/10.1023/A:1010933404324