Maternal Health Risk Prediction Using Machine Learning

Tasmita Tanjim

Student ID: 19723

Project Overview

Overview

- Maternal health is vital for both mothers and newborns. Early identification of risk can help save lives.
- This project uses machine learning to predict maternal health risk levels—high, mid, or Low—based on various features.

Problem

- Traditional risk detection relies on subjective judgment, which can delay accurate and timely diagnosis.

Solution

- Built Random Forest and XGBoost models to classify maternal risk levels automatically.

Dataset Description

- Source: UCI Machine Learning Repository
- Total Records: 1014
- Data Shape: (1014, 7)
- Target Variable: RiskLevel (High, Mid, Low)
- Feature Details: 6 numerical health indicators
  - Age: Age of the pregnant woman (in years)
  - SystolicBP: Upper blood pressure (mmHg)
  - DiastolicBP: Lower blood pressure (mmHg)
  - BS: Blood sugar level (mmol/L)
  - BodyTemp: Body temperature
  - HeartRate: Heart rate (beats per minute)
- No missing values in the dataset
- Problem Type: Classification

```
data.size
```

✓ 0.0s

```
7098
```

```
missing_values = data.isnull().sum()
print(missing_values)
```

✓ 0.0s

```
Age            0
SystolicBP     0
DiastolicBP    0
BS             0
BodyTemp       0
HeartRate      0
RiskLevel      0
dtype: int64
```

```
unique_levels = data['RiskLevel'].unique()
print("Unique Levels in 'RiskLevel':", unique_levels)
```

✓ 0.0s

```
Unique Levels in 'RiskLevel': ['high risk' 'low risk' 'mid risk']
```

Project Workflow

1. Data Collection

→ Collected dataset from UCI ML Repository

2. Data Preprocessing

→ Cleaned data, handled outliers

→ Encoded labels, scaled features

3. Exploratory Data Analysis (EDA)

→ Visualized feature distributions and correlations

→ Identified most influential features

4. Model Selection

→ Trained 5 models: DT, KNN, SVM, RF, XGBoost

→ Evaluated using cross-validation

5. Hyperparameter Tuning

→ Applied GridSearchCV on RF & XGBoost

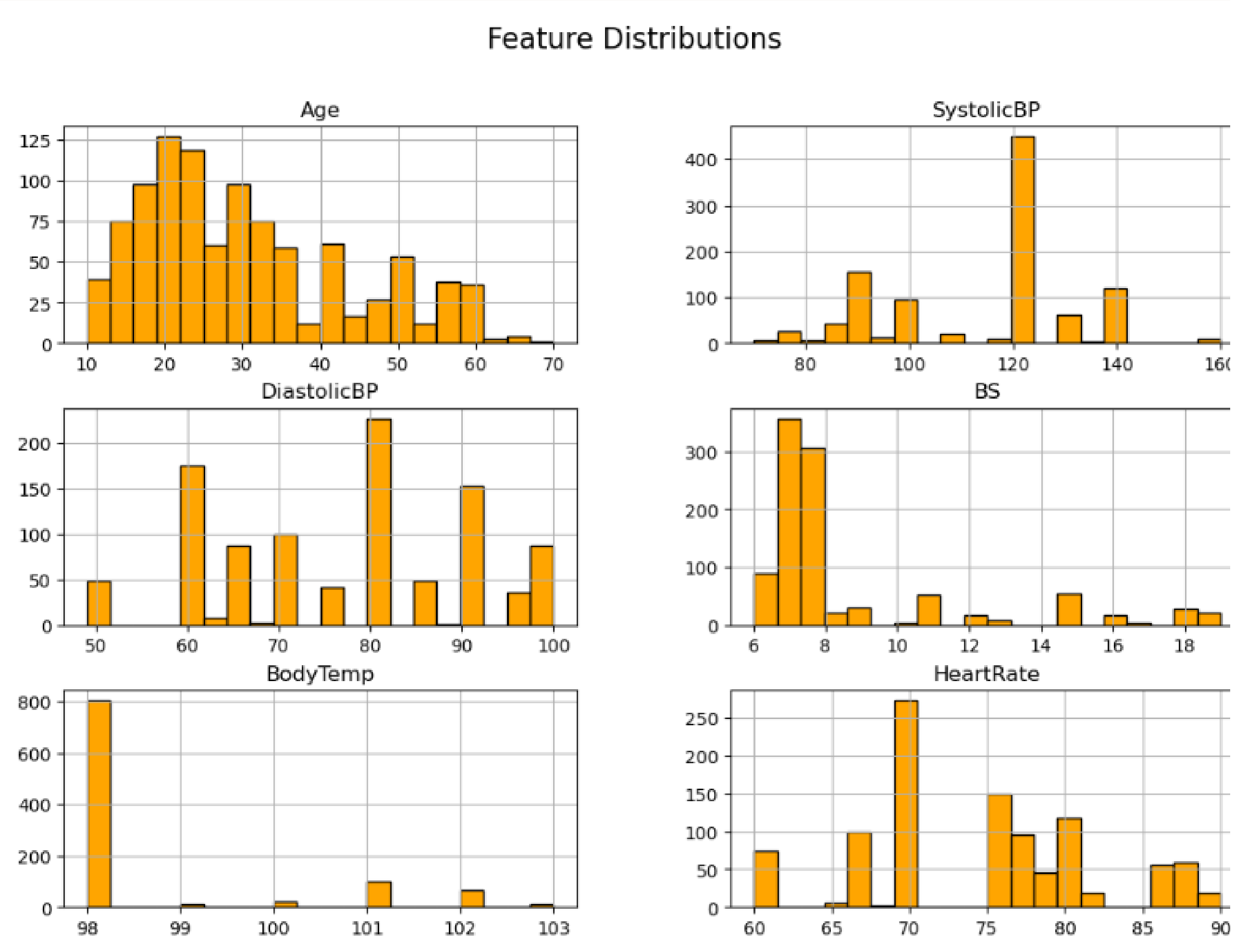
→ Found best-performing parameters

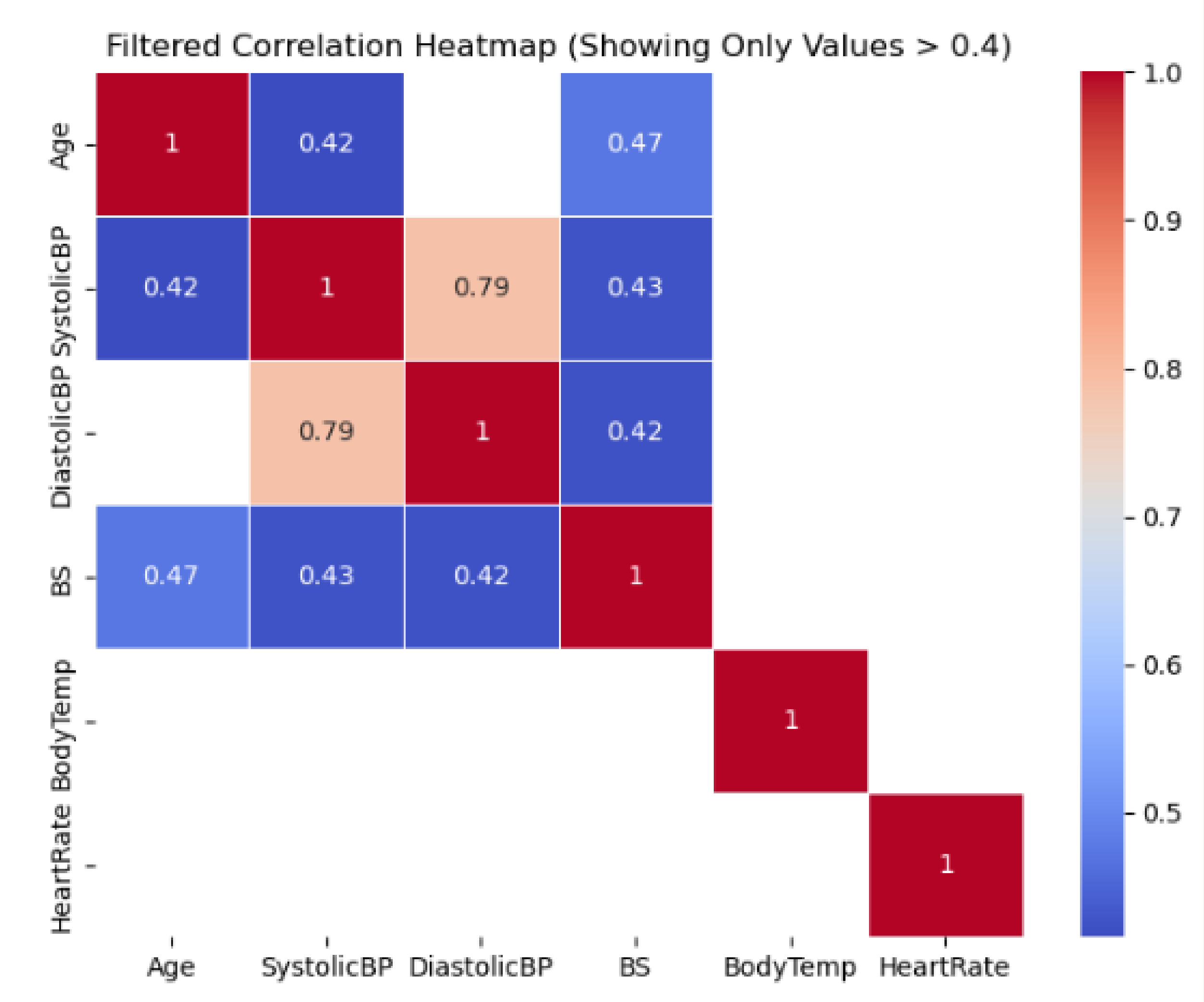
6. Model Evaluation & Interpretation

→ Accuracy, classification report, confusion matrix

→ Identified feature importance for interpretability

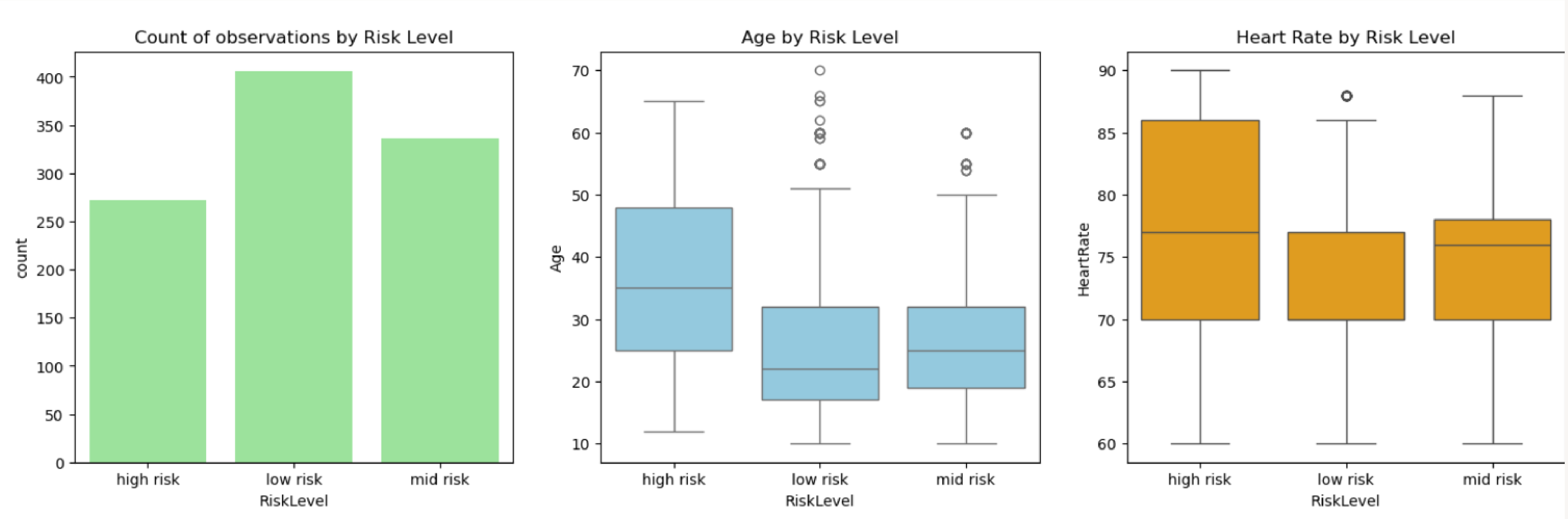
Data Visualization





- Most women are pregnant at 20-40 years, sharp range from 20-30.
- The dataset seems well distributed, and we need to perform scaling
- SystolicBP & DiastolicBP (0.79 correlation) => Strong correlation
- Removing DiastolicBP significantly lowered Random Forest's performance.

Risk Level Analysis



- Low risk has the highest number of cases and high risk has the fewest.
- Age between 26-48 seems to have higher pregnancy risk.
- Higher heart-beat is associated with high-risk pregnancy.

Data Preprocessing (Encoding)

```
print("Before encoding 'RiskLevel'(target val
```

✓ 0.0s

```
Before encoding 'RiskLevel'(target value & categ
 0        high risk
1        high risk
2        high risk
3        high risk
4         low risk
            ...
1009     high risk
1010     high risk
1011     high risk
1012     high risk
1013      mid risk
Name: RiskLevel, Length: 1014, dtype: object
```

```
encoder = LabelEncoder()
data['RiskLevel'] = encoder.fit_transform(data['RiskLevel'])
print("'RiskLevel' encoded successfully!")
print("After encoding 'RiskLevel'(target value & categorical
```

✓ 0.0s

```
'RiskLevel' encoded successfully!
After encoding 'RiskLevel'(target value & categorical feature)
 0       0
1        0
2        0
3        0
4        1
        ..
1009     0
1010     0
1011     0
1012     0
1013     2
Name: RiskLevel, Length: 1014, dtype: int32
```

Feature Scaling (StandardScaler)

Applied Train-validation the dataset(80% training & 20% **validation)**

Before Scaling: Age = 17, 35, 12
After Scaling: Age = -0.95, 0.38, -1.32
Now, values are adjusted relative to the mean

```python
print("Before scaling:\n",X_train.head(6))
```

0.0s

Before scaling:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
992	17	110	75	13.0	101.0	76
883	35	120	60	6.1	98.0	76
251	12	95	60	6.9	98.0	65
294	29	130	70	7.7	98.0	78
756	23	130	70	6.9	98.0	70
582	19	120	76	7.5	98.0	66

```python
# Scale the numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
```

0.0s

```python
#Converting X_train_scaled into a dataFrame with correct column na
X_train = pd.DataFrame(X_train_scaled, columns=X.columns)
X_val = pd.DataFrame(X_val_scaled, columns=X.columns)
print('Feature Scaling is done here.')
print(X_train.head())
```

0.0s

Feature Scaling is done here.

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
0	-0.955493	-0.163860	-0.098968	1.292816	1.671023	0.195731
1	0.380806	0.371557	-1.173914	-0.798682	-0.492680	0.195731
2	-1.326687	-0.966986	-1.173914	-0.556189	-0.492680	-1.258153
3	-0.064627	0.906974	-0.457283	-0.313697	-0.492680	0.460073
4	-0.510060	0.906974	-0.457283	-0.556189	-0.492680	-0.597297

Model Selection & Training

```python
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(),
    "XGBoost": XGBClassifier()
}
cv_folds = 5
model_results = []
for model_name, model in models.items():
    scores = cross_val_score(model, X_train, y_train, cv=cv_folds,
                             scoring='accuracy', n_jobs=-1)
    result = {
        "Model": model_name,
        "Mean Accuracy": round(scores.mean(), 4),
        "Standard Deviation": round(scores.std(), 4)
    }
    model_results.append(result)

cv_results_df = pd.DataFrame(model_results)
cv_results_df = cv_results_df.sort_values(by='Mean Accuracy',ascending=False)

print("Final Model Comparison:")
print(cv_results_df)
```

```
Final Model Comparison:
           Model  Mean Accuracy  Standard Deviation
1  Random Forest         0.8274              0.0174
4        XGBoost         0.8249              0.0167
0  Decision Tree         0.8151              0.0258
3            SVM         0.6991              0.0176
2            KNN         0.6769              0.0085
```

Random Forest (0.8274) performed the best, followed by XGBoost (0.8249).

Decision Tree had slightly lower accuracy (0.8151) but a higher STD(0.0258), meaning it is less stable.

Hyperparameter Tuning

```python
rf_params = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

#Setting parameters to find best combination using GridSearchCV
rf_grid = GridSearchCV(RandomForestClassifier(random_state=42),
                       rf_params,cv=5, scoring='accuracy', n_jobs=-1)

rf_grid.fit(X_train, y_train)
print(f"Best parameters (Random Forest): {rf_grid.best_params_}")
print(f"Best accuracy (Random Forest): {round(rf_grid.best_score_, 4)}")
```

```python
xgb_params = {
    'n_estimators': [50, 100, 150],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.7, 0.8, 1.0]
}
# Tuning XGBoost with GridSearch
xgb_grid = GridSearchCV(XGBClassifier(random_state=42),
                        xgb_params, cv=5, scoring='accuracy', n_jobs=-1)

xgb_grid.fit(X_train, y_train)
print(f"Best parameters (XGBoost): {xgb_grid.best_params_}")
print(f"Best accuracy (XGBoost): {round(xgb_grid.best_score_, 4)}")
```

```
Best accuracy (XGBoost): 0.8311
```

```
Best parameters (Random Forest): {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Best accuracy (Random Forest): 0.8286
```

Improvement in Random Forest Accuracy: (82.86% - 82.74%) = 0.12% increase
Improvement in XGBoost Accuracy: (83.11% - 82.49%) = 0.62% increase
After tuning, XGBoost is the best model, and then Random Forest

Conclusion

```
Final Evaluation: Random Forest Model
Accuracy: 0.867

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95        55
           1       0.90      0.81      0.86        81
           2       0.76      0.87      0.81        67

    accuracy                           0.87       203
   macro avg       0.88      0.88      0.87       203
weighted avg       0.87      0.87      0.87       203

Confusion Matrix:
 [[52  0  3]
 [ 0 66 15]
 [ 2  7 58]]
```

```
Final Evaluation: XGBoost Model
Accuracy: 0.8621

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.91      0.93        55
           1       0.92      0.81      0.86        81
           2       0.75      0.88      0.81        67

    accuracy                           0.86       203
   macro avg       0.88      0.87      0.87       203
weighted avg       0.87      0.86      0.86       203

Confusion Matrix:
 [[50  0  5]
 [ 0 66 15]
 [ 2  6 59]]
```

Model Name	Initial Accuracy	Accuracy after tuning	Accuracy after training with best params
Random Forest	0.8274	0.8286	0.8274(4.66% up)
XGBoost	0.8249	0.8311	0.8249(4.51% up)

Thank you!