Fetal Health Classification

Project Description:

The Fetal Health Classification project aims to develop a machine learning model to classify the outcomes of Cardiotocography (CTG) tests, a cost-effective diagnostic tool used during pregnancy to monitor fetal heart rate (FHR) and uterine contractions. By leveraging a dataset of 2126 records labelled by expert obstetricians into Normal, Suspect, and Pathological categories, the model will analyse key features such as FHR variability, uterine contractions, and histogram data to predict fetal health outcomes. This project seeks to enhance early detection of fetal distress, enabling timely medical interventions to improve maternal and child health outcomes while demonstrating the transformative potential of machine learning in healthcare.

Approach:

The approach for the Fetal Health Classification project involves leveraging machine learning models to classify fetal health outcomes using Cardiotocography (CTG) data. The process begins with data preprocessing, including handling missing values, normalizing features, and splitting the dataset for training and testing. Feature analysis will be conducted to understand the importance of variables like fetal heart rate variability, uterine contractions, and decelerations. Machine learning algorithms such as Linear Regression, Logistic Regression, K-Nearest Neighbors (KNN), Decision Tree, and K-Means Clustering will be implemented to build predictive models. Each model's performance will be evaluated using metrics like accuracy, precision, recall, and F1 score to identify the most effective approach. The chosen model will then be fine-tuned to enhance its predictive capability, ensuring reliable classification of fetal health states.

The dataset consists of 2126 records from Cardiotocography (CTG) tests, classified into three categories: Normal, Suspect, and Pathological. It includes features such as fetal heart rate, uterine contractions, and fetal movements, which help assess fetal health.

The target variable, fetal_health, is labelled as

1: Normal

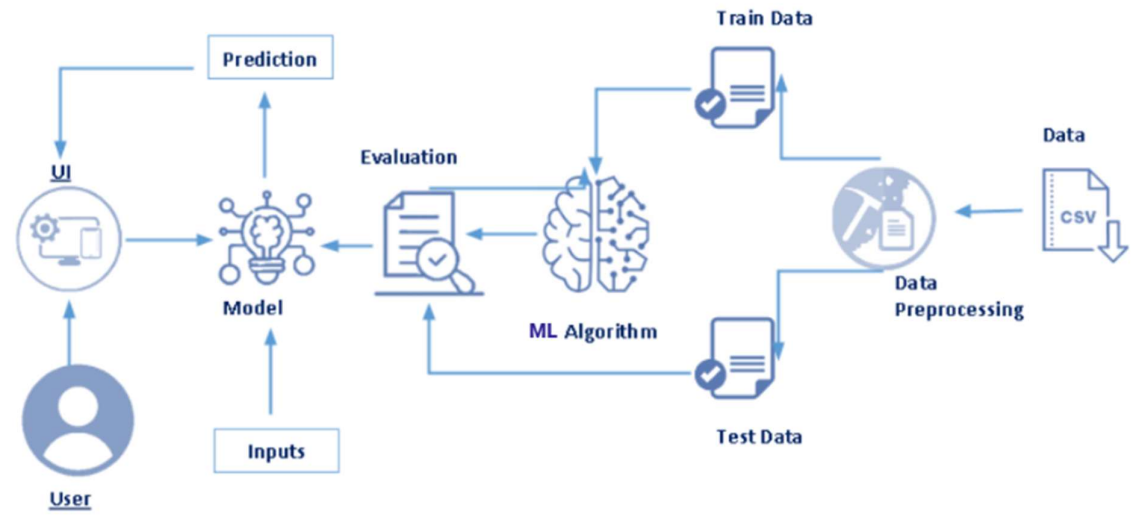
2: Suspect

3: Pathological

A sample of the dataset with 10 rows is shown below. This dataset is used to train machine learning models to predict fetal health outcomes.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	baseline v	acceleratic	fetal_mov	uterine_cc	light_dece	severe_de	prolongue	abnormal_	mean_vali	percentag	mean_vali	histogram	histogram	histogram	histogram	histogram	histogram	histogram	histogram	histogram	histogram	fetal_health
2	120	0	0	0	0	0	0	73	0.5	43	2.4	64	62	126	2	0	120	137	121	73	1	2
3	132	0.006	0	0.006	0.003	0	0	17	2.1	0	10.4	130	68	198	6	1	141	136	140	12	0	1
4	133	0.003	0	0.008	0.003	0	0	16	2.1	0	13.4	130	68	198	5	1	141	135	138	13	0	1
5	134	0.003	0	0.008	0.003	0	0	16	2.4	0	23	117	53	170	11	0	137	134	137	13	1	1
6	132	0.007	0	0.008	0	0	0	16	2.4	0	19.9	117	53	170	9	0	137	136	138	11	1	1
7	134	0.001	0	0.01	0.009	0	0.002	26	5.9	0	0	150	50	200	5	3	76	107	107	170	0	3
8	134	0.001	0	0.013	0.008	0	0.003	29	6.3	0	0	150	50	200	6	3	71	107	106	215	0	3
9	122	0	0	0	0	0	0	83	0.5	6	15.6	68	62	130	0	0	122	122	123	3	1	3
10	122	0	0	0.002	0	0	0	84	0.5	5	13.6	68	62	130	0	0	122	122	123	3	1	3

Technical Architecture:



Prerequisites:

To complete the Fetal Health Classification project, the following software, packages, and concepts are required:

1. **Google Collab:**

   A free cloud-based platform for running Python code in a Jupyter notebook environment, ideal for machine learning tasks with access to GPUs for faster model training.

2. **Machine Learning Libraries:**

   - **Numpy**: For numerical computations and handling arrays and matrices.
   - **Pandas:** For data manipulation and analysis.
   - **Scikit-learn**: For implementing machine learning models such as Linear Regression, Logistic Regression, K-Nearest Neighbors (KNN), K-Means Clustering, and Decision Trees.

3. **Machine Learning Concepts:**

   Basic understanding of machine learning algorithms, including Linear Regression, Logistic Regression, K-Nearest Neighbors, K-Means Clustering, and Decision Trees.

4. **Dataset:**

   The Fetal Health Classification dataset, which contains records of features from Cardiotocogram (CTG) exams. The dataset is available for download from Kaggle.
   Link: https://www.kaggle.com/code/karnikakapoor/fetal-health-classification

Project Objectives:

1. Develop a machine learning model to classify fetal health using Cardiotocogram (CTG) data.

2. Monitor fetal well-being and detect potential distress during pregnancy.

4. Classify CTG test outcomes into three categories: Normal, Suspect, and Pathological.

5. Assist healthcare professionals in assessing pregnancy risk levels and taking necessary actions for the safety of the mother and fetus.

Project Flow:

1. Data Collection:

CTG Data: Collect the dataset from Kaggle, containing 2126 records with labelled outcomes: Normal, Suspect, and Pathological. The dataset includes features like FHR, uterine contractions, and fetal movements.

2. Data Preprocessing:

- Clean the uploaded dataset (handle missing values).
- Apply scaling and normalization to the features.
- Perform feature extraction and transformation to prepare the data for model training.

3. Model Building:

- Build and configure various machine learning models (Linear Regression, Logistic Regression, KNN, K-Means, Decision Trees).
- Split the data into training and testing sets.
- Train the model using the training data and evaluate it using the testing data.

4. Model Evaluation:

- Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.
- Tune hyperparameters if necessary to improve performance.

5. Prediction:

- Use the trained model to predict the fetal health status (Normal, Suspect, or Pathological) based on the provided CTG features.

6. Results Display:

- Display the predicted result (Normal, Suspect, or Pathological) .

Project Structure:

Milestone 1: Data Collection

- CTG Data: Collect the dataset from Kaggle, containing 2126 records with labeled outcomes: Normal, Suspect, and Pathological. The dataset includes features like FHR, uterine contractions, and fetal movements.
- File Structure:
    - fetal_health.csv – The main data file for training models.

Milestone 2: Data Preprocessing

1. Displayed the dataset's first and last rows (head and tail), information (info()), summary statistics (describe()), and all column names to analyse structure, data types, non-null values, and feature distributions.

```
[ ] df.columns

Index(['baseline value', 'accelerations', 'fetal_movement',
       'uterine_contractions', 'light_decelerations', 'severe_decelerations',
       'prolongued_decelerations', 'abnormal_short_term_variability',
       'mean_value_of_short_term_variability',
       'percentage_of_time_with_abnormal_long_term_variability',
       'mean_value_of_long_term_variability', 'histogram_width',
       'histogram_min', 'histogram_max', 'histogram_number_of_peaks',
       'histogram_number_of_zeroes', 'histogram_mode', 'histogram_mean',
       'histogram_median', 'histogram_variance', 'histogram_tendency',
       'fetal_health'],
      dtype='object')
```

2. Counted the occurrences of each category in the target variable (fetal_health) using value_counts() to understand class distribution.

```
df['fetal_health'].value_counts()
# 1 - Normal 2 - Suspect 3 - Pathological
```

fetal_health	count
1.000000	1655
2.000000	295
3.000000	176

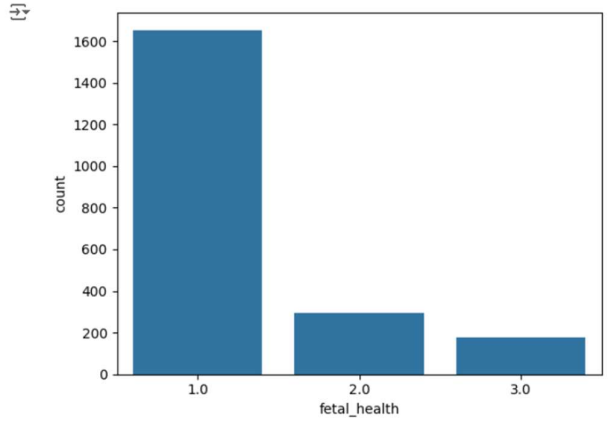
dtype: int64

3. Identified missing values with isnull() and handled them using mean or median imputation.

4. Plotted the distribution of target values using a count plot to understand class imbalance in fetal_health.

```
import seaborn as sns
import matplotlib.pyplot as plt
# count plot on single categorical variable
sns.countplot(df,x ='fetal_health')

# Show the plot
plt.show()
```



5. Separated features (X) and target (y), dropping fetal_health from predictors.

```
[ ]  x=df.drop(['fetal_health'],axis=1)
     y=df['fetal_health']
     x.head()
```

6. Standardized numerical features using StandardScaler for normalization.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
```

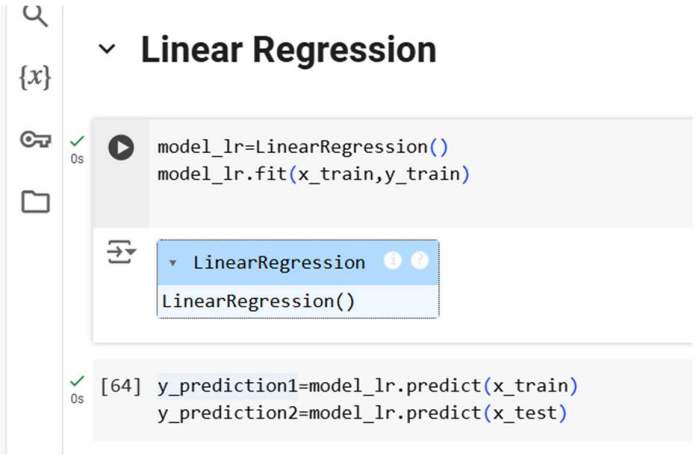
7. Cleaned the dataset to ensure quality and consistency for model training.

Milestone 3: Model Building

1. Imported essential libraries including numpy, pandas, seaborn, matplotlib, and sklearn for data manipulation, visualization, and model building.

```
[2] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.linear_model import LinearRegression, LogisticRegression
    from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, confusion_matrix, classification_report, recall_score, precision_score, f1_score
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.cluster import KMeans
    from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score, homogeneity_score, completeness_score, v_measure_score
```

2. Used models such as Linear Regression, Logistic Regression, Decision Tree Classifier, KNeighbors Classifier, and KMeans for training.



Logistic Regressison

```
model_lr1=LogisticRegression(max_iter=12000)
model_lr1.fit(x_train,y_train)
y_prediction_lr1=model_lr1.predict(x_train)
y_prediction_lr2=model_lr1.predict(x_test)
```

```
print(y_prediction_lr1)
print(y_prediction_lr2)
#The ellipsis (...) indicates that some elements in the array are omitted in the printed output.
```

```
[1. 1. 1. ... 1. 1. 1.]
[1. 1. 1. 1. 1. 3. 1. 3. 1. 2. 1. 3. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 3. 1.
 1. 3. 3. 2. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 3. 3. 2. 1. 2. 1. 1. 1. 2. 1. 2.
 1. 1. 1. 1. 2. 1. 2. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 3. 1.
 1. 2. 2. 1. 2. 1. 1. 1. 1. 2. 1. 1. 3. 1. 2. 2. 1. 2. 1. 3. 1. 1. 2. 3.
 1. 1. 1. 1. 3. 3. 3. 1. 1. 1. 3. 1. 1. 1. 1. 1. 1. 1. 2. 2. 2. 2. 3. 1.]
```

Decision Tree

```
model_dt=DecisionTreeClassifier(criterion='entropy')
model_dt.fit(x_train,y_train)
y_prediction_train=model_dt.predict(x_train)
y_prediction_test=model_dt.predict(x_test)
```

K-Nearest Neighbour

```
model_knn=KNeighborsClassifier(n_neighbors=5)
model_knn.fit(x_train,y_train)
y_prediction_train_knn=model_knn.predict(x_train)
y_prediction_test_knn=model_knn.predict(x_test)
print(y_prediction_train_knn)
print(y_prediction_test_knn)
```

```
[1. 1. 1. ... 1. 1. 1.]
[1. 1. 1. 1. 1. 3. 1. 3. 2. 2. 1. 3. 1. 1. 1. 1. 1. 1. 1. 1. 1. 3. 1.
 1. 3. 3. 1. 1. 1. 2. 1. 1. 2. 1. 1. 1. 3. 2. 1. 1. 2. 1. 1. 1. 2. 1. 2.
 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1.
 1. 1. 1. 2. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 2. 1.
 1. 2. 1. 1. 2. 1. 1. 1. 1. 2. 1. 1. 3. 1. 2. 2. 1. 2. 1. 1. 1. 1. 1. 2.
 1. 2. 1. 1. 1. 1. 3. 1. 1. 1. 3. 1. 1. 1. 2. 1. 1. 1. 2. 2. 2. 1. 2. 1.
 3. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 2. 1. 2. 2. 1.
 2. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 3. 2. 2. 1.]
```

K Means Clustering

```
k=3
model_kmeans=KMeans(n_clusters=k,random_state=42)
numerical_features = df.select_dtypes(include=['number']).drop(columns=['fetal_health'])
df_for_clustering = df[numerical_features.columns]
model_kmeans.fit(df_for_clustering)
cluster_labels=model_kmeans.labels_
cluster_centres=model_kmeans.cluster_centers_
plt.figure(figsize=(8,6))
plt.scatter(df_for_clustering.iloc[:, 0], df_for_clustering.iloc[:, 1], c=cluster_labels, cmap='viridis', alpha=0.7)
plt.scatter(model_kmeans.cluster_centers_[:, 0], model_kmeans.cluster_centers_[:, 1], marker='x', s=200, color='red', label='Centroids')
plt.title('K-Means Clustering')
plt.xlabel(df_for_clustering.columns[0])  # Label for the first selected feature
plt.ylabel(df_for_clustering.columns[1])  # Label for the second selected feature
plt.legend()
plt.show()
```

Milestone 4: Model Evaluation

- Evaluated model performance using metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared Score (R^2), Accuracy, Precision, Recall, F1-Score, and clustering scores (Davies-Bouldin Index, Calinski-Harabasz Index).

LINEAR REGRESSION

```
Mean Squared Error (MSE) on train data : 0.14949950075233076
R² Score on train data : 0.622516897563943
Root Mean Squared Error (RMSE) on train data : 0.3866516529802126
Mean Squared Error (MSE) on test data : 0.15241553117145193
R² Score on test data : 0.5424150325624352
Root Mean Squared Error (RMSE) : 0.39040431756251354
```

LOGISTIC REGRESSION

```
Confusion Matrix:
 [[471  19   6]
 [ 31  62   8]
 [  1   4  36]]
shape of confusion matrix: (3, 3)
Multi-class classification. TP, TN, FP, FN not directly applicable.
Recall of Logistic Reg on test data : 0.8918495297805643
Precision of Logistic Reg on test data : 0.8897114675032965
F1_Score of Logistic Reg on test data : 0.8894554756623722
```

K-Nearest Neighbors (KNN)

```
Accuracy of KNN on test data 0.8761755485893417
Precision of KNN on test data 0.871369346286609
Recall of KNN on test data 0.8761755485893417
F1_Score of KNN on test data 0.8732285111392847
Accuracy of KNN on train data 0.9240591397849462
Precision of KNN on train data 0.9210023006978251
Recall of KNN on train data 0.9240591397849462
F1_Score of KNN on train data 0.9211870197390722
```

K-MEANS CLUSTERING

```
Inertia: 6110135.472979927
Silhouette Score: -6110135.472979927
Davies-Bouldin Score: 1.0532223130703062
Calinski-Harabasz Score: <function calinski_harabasz_score at 0x7ce9348ab370>
Homogenity: 0.034250311501843586
Completeness: 0.03507636009354344
V-Measure: 0.03465841448631258
<matplotlib.collections.PathCollection at 0x7ce92cb4dc00>
```

Comparison of Training and Test Accuracy Across Models

	Model	Score
1	Linear Regression (Test)	0.542415
0	Linear Regression (Train)	0.622517
2	Logistic Regression (Train)	0.875672
7	KNN (Test)	0.876176
3	Logistic Regression (Test)	0.891850
5	Decision Tree (Test)	0.921630
6	KNN (Train)	0.924059
4	Decision Tree (Train)	1.000000

Milestone 5: Model Prediction

- Predictions are made using five models: Logistic Regression, Decision Tree, K-Nearest Neighbors, and K-Means clustering.
- Each model provides a prediction for fetal health status (Normal, Suspect, or Pathological).
- The results from all models are printed to assess the predictions.

```
new_features = [132.000000, 0.007000, 0.000000, 0.008000, 0.000000, 0.000000, 0.000000, 16.0
# Reshape new_features to a 2D array with one row and multiple columns
new_features = np.array(new_features).reshape(1, -1)

predicted_health = model_lr.predict(new_features)
print("predicted fetal health is ", predicted_health)

predicted_health = model_lr1.predict(new_features)
print("predicted fetal health is ", predicted_health)

predicted_health = model_dt.predict(new_features)
print("predicted fetal health is ", predicted_health)

predicted_health = model_knn.predict(new_features)
print("predicted fetal health is ", predicted_health)

predicted_health = model_kmeans.predict(new_features)
print("predicted fetal health is ", predicted_health)
```

```
predicted fetal health is  [0.74172928]
predicted fetal health is  [1.]
predicted fetal health is  [1.]
predicted fetal health is  [1.]
predicted fetal health is  [1]
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but KMeans was fitted with feature names
  warnings.warn(
```