BHARATIYA VIDYA BHAVAN'S

SARDAR PATEL INSTITUTE OF TECHNOLOGY

(Empowered Autonomous Institute Affiliated to University of Mumbai)

[Knowledge is Nectar]

Department of Computer Engineering

Course - Artificial Intelligence for Healthcare(AIH)

UID	2021300101
Name	Adwait Purao
Class and Batch	BE Computer Engineering - Batch D
Date	14/8/24
Lab #	1 - Regression in Healthcare Dataset
Objective	● Write a program for regression analysis for healthcare dataset. ● To demonstrate the working principle of regression techniques on medical data set for building the model to classify/ predict using a new sample.
Outcomes	● Explore the Medical Dataset suitable for linear/ logistic regression problem ● Explore the pattern from the dataset and apply suitable algorithm
Theory	**What is regression with a mathematical approach?** Regression analysis is a statistical method used to model and analyze the relationships between variables. It helps us understand how the dependent variable changes when any one of the independent variables is varied while the other independent variables are held fixed. The most common form of regression is linear regression, where the relationship between variables is modeled as a linear equation. 1. **Linear Regression** Linear regression aims to find the best-fitting straight line through the data points. The equation of a simple linear regression line is given by: $y=\beta 0 +\beta 1 x+\epsilon$ where: ● Y is the dependent variable. ● X is the independent variable. ● β0 is the y-intercept of the regression line. ● β1 is the slope of the regression line. ● ϵ is the error term, representing the difference between the observed and predicted values. **Multiple Linear Regression** extends this concept to include multiple independent variables: $y=\beta 0 +\beta 1 x1 +\beta 2 x2 +\ldots+\beta n xn +\epsilon$ where: ● x1 ,x2 ,…,xn are the independent variables.

● $\beta_1,\beta_2,\ldots,\beta_n$ are the coefficients representing the impact of each independent variable on the dependent variable.

2. **Objective of Regression**

The objective of regression analysis is to estimate the coefficients ($\beta_0,\beta_1,\ldots,\beta_n$) such that the
sum of squared differences between the observed and predicted values is minimized. This is
known as the least squares method.
Mathematically, this is represented as:
where:
● y_i is the observed value.
● $\hat{y}_i$ is the predicted value from the regression model.
● M is the number of observations.

3. **Assumptions of Linear Regression**

For linear regression to provide reliable results, certain assumptions must be satisfied:
● **Linearity**: The relationship between the independent and dependent variables is linear.
● **Independence:** Observations are independent of each other.
● **Homoscedasticity**: The variance of error terms is constant across all levels of the independent variables.
● **Normality**: The residuals (errors) of the model are normally distributed.

4. **Interpretation of Coefficients**

● **Intercept (β_0)**: Represents the expected mean value of y when all x variables are zero.
● **Slope ($\beta_1,\beta_2,\ldots,\beta_n$)**: Represents the change in the mean value of y for a one-unit change
in the respective x variable, holding all other variables constant.

5. **Goodness of Fit**

measures the proportion of variability in the dependent variable that can be explained by the
independent variables.
where:
● $\bar{y}$ is the mean of the observed values.
A higher R^2 value indicates a better fit of the model to the data.

6. **Non-Linear Regression**

When the relationship between variables is not linear, non-linear regression models can be used. These models fit the data using a nonlinear function, such as polynomial, exponential, or
logarithmic functions.

7. **Applications**

Regression analysis is widely used in various fields such as finance (to predict stock prices),

economics (to estimate demand curves), biology (to analyze growth patterns), and many more
areas where relationships between variables need to be understood and quantified.
What are the types of regression and its significance?

1. **Linear Regression**

○ **Simple Linear Regression**: Models the relationship between two variables using a linear equation. The model is expressed as:
$y=\beta_0+\beta_1 x+\epsilon$ where y is the dependent variable, x is the independent variable, β_0 is the y-intercept, β_1 is the slope, and ϵ is the error term.

○ **Multiple Linear Regression**: Extends simple linear regression by including multiple independent variables:
$y=\beta_0+\beta_1x_1+\beta_2x_2+\ldots+\beta_nx_n+\epsilon$

Significance:

○ **Prediction:** Linear regression is widely used for predicting the value of the dependent variable based on independent variables.

○ **Relationship Analysis**: Helps in understanding and quantifying the strength and direction of relationships between variables.

Simplicity and Interpretability: Provides a straightforward approach that is easy to interpret and apply to real-world problems.

2. **Polynomial Regression**

○ Models a non-linear relationship between the independent and dependent variables by including polynomial terms:

$$y=\beta_0+\beta_1x+\beta_2x2+\ldots+\beta_nxn+\epsilon$$

Significance:

○ **Flexibility:** Suitable for modeling curvilinear data trends that linear regression cannot capture.

○ **Capturing Complexity:** Allows fitting complex data patterns without the need for advanced machine learning techniques.

3. **Logistic Regression**

○ Used for binary classification problems where the dependent variable is categorical (e.g., yes/no, true/false). The logistic regression model estimates the probability of a class occurrence using the logistic function:

$$P(y=1|x)=\frac{1}{1+e^{-(\beta_0+\beta_1x_1+\ldots+\beta_nx_n)}}$$

Significance:

○ **Classification:** Effective for binary and multi-class classification tasks, predicting probabilities of class membership.

○ **Odds Ratio Interpretation:** Provides insights into the impact of predictors on the likelihood of outcomes.

○ **Wide Applicability**: Used in fields like medicine, finance, and marketing to

	model binary outcomes. 4. **Ridge and Lasso Regression** ○ **Ridge Regression:** Adds a penalty term proportional to the square of the coefficients to the linear regression cost function, helping to address multicollinearity and overfitting: $$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$ ○ **Lasso Regression:** Similar to ridge regression but uses an absolute value penalty, which can shrink some coefficients to zero, effectively performing variable Selection: $$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \lvert\beta_j\rvert$$ **Significance:** ○ **Feature Selection:** Lasso regression helps in selecting important variables, simplifying models. ○ **Handling Multicollinearity:** Ridge regression stabilizes estimates when predictors are highly correlated. ○ Regularization: Both methods prevent overfitting by constraining coefficient sizes. **Significance of Regression Analysis** ● **Prediction and Forecasting:** Regression models provide valuable tools for predicting future outcomes based on historical data, aiding in decision-making across various fields. ● **Understanding Relationships:** Helps quantify and understand the relationships between variables, providing insights into causal or associative links. ● **Model Simplicity:** Linear and logistic regression offer simple yet powerful models that are easy to interpret and apply. ● **Data-Driven Decisions:** Enables businesses and researchers to make informed decisions by identifying and analyzing key factors that affect outcomes.
Implementation / Code	**Logistic Regression:** Dataset: https://www.kaggle.com/code/karnikakapoor/fetal-health-classification **ALGORITHM:** Step 1: Create a sample dataset with multiple independent variables and one dependent

variable (Y).
Step 2: The data is split into training and testing sets using the train_test_split function.
Step3: Regression model is created and fitted to the training data.
Step4: Predictions are made on the test set.
Step5: The model is evaluated using metrics like Accuracy, F1 Score, Precision, Recall.

Code:

```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_score, recall_score,
confusion_matrix, classification_report, accuracy_score, f1_score
from sklearn import metrics
from sklearn.metrics import roc_curve, auc, roc_auc_score
np.random.seed(0)
data = pd.read_csv("./fetal_health.csv")
data.head()
```

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term_variability
0	120.0	0.000	0.0	0.000	0.000	0.0	0.0	73.0
1	132.0	0.006	0.0	0.006	0.003	0.0	0.0	17.0
2	133.0	0.003	0.0	0.008	0.003	0.0	0.0	16.0
3	134.0	0.003	0.0	0.008	0.003	0.0	0.0	16.0
4	132.0	0.007	0.0	0.008	0.000	0.0	0.0	16.0

5 rows × 22 columns

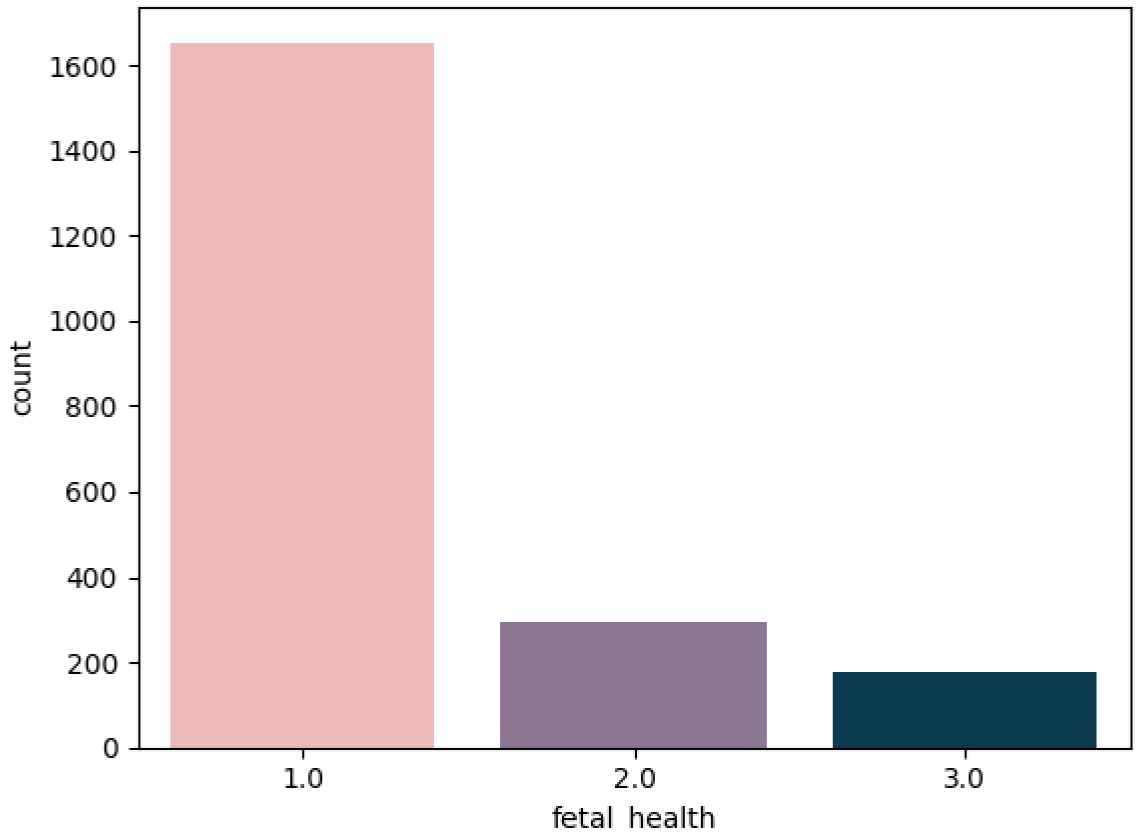
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype  
---  ------                                                  --------------  -----  
 0   baseline value                                          2126 non-null   float64
 1   accelerations                                           2126 non-null   float64
 2   fetal_movement                                          2126 non-null   float64
 3   uterine_contractions                                    2126 non-null   float64
 4   light_decelerations                                     2126 non-null   float64
 5   severe_decelerations                                    2126 non-null   float64
 6   prolongued_decelerations                                2126 non-null   float64
 7   abnormal_short_term_variability                         2126 non-null   float64
 8   mean_value_of_short_term_variability                    2126 non-null   float64
 9   percentage_of_time_with_abnormal_long_term_variability  2126 non-null   float64
 10  mean_value_of_long_term_variability                     2126 non-null   float64
 11  histogram_width                                         2126 non-null   float64
 12  histogram_min                                           2126 non-null   float64
 13  histogram_max                                           2126 non-null   float64
 14  histogram_number_of_peaks                               2126 non-null   float64
 15  histogram_number_of_zeroes                              2126 non-null   float64
 16  histogram_mode                                          2126 non-null   float64
 17  histogram_mean                                          2126 non-null   float64
 18  histogram_median                                        2126 non-null   float64
 19  histogram_variance                                      2126 non-null   float64
 20  histogram_tendency                                      2126 non-null   float64
 21  fetal_health                                            2126 non-null   float64
dtypes: float64(22)
memory usage: 365.5 KB
```

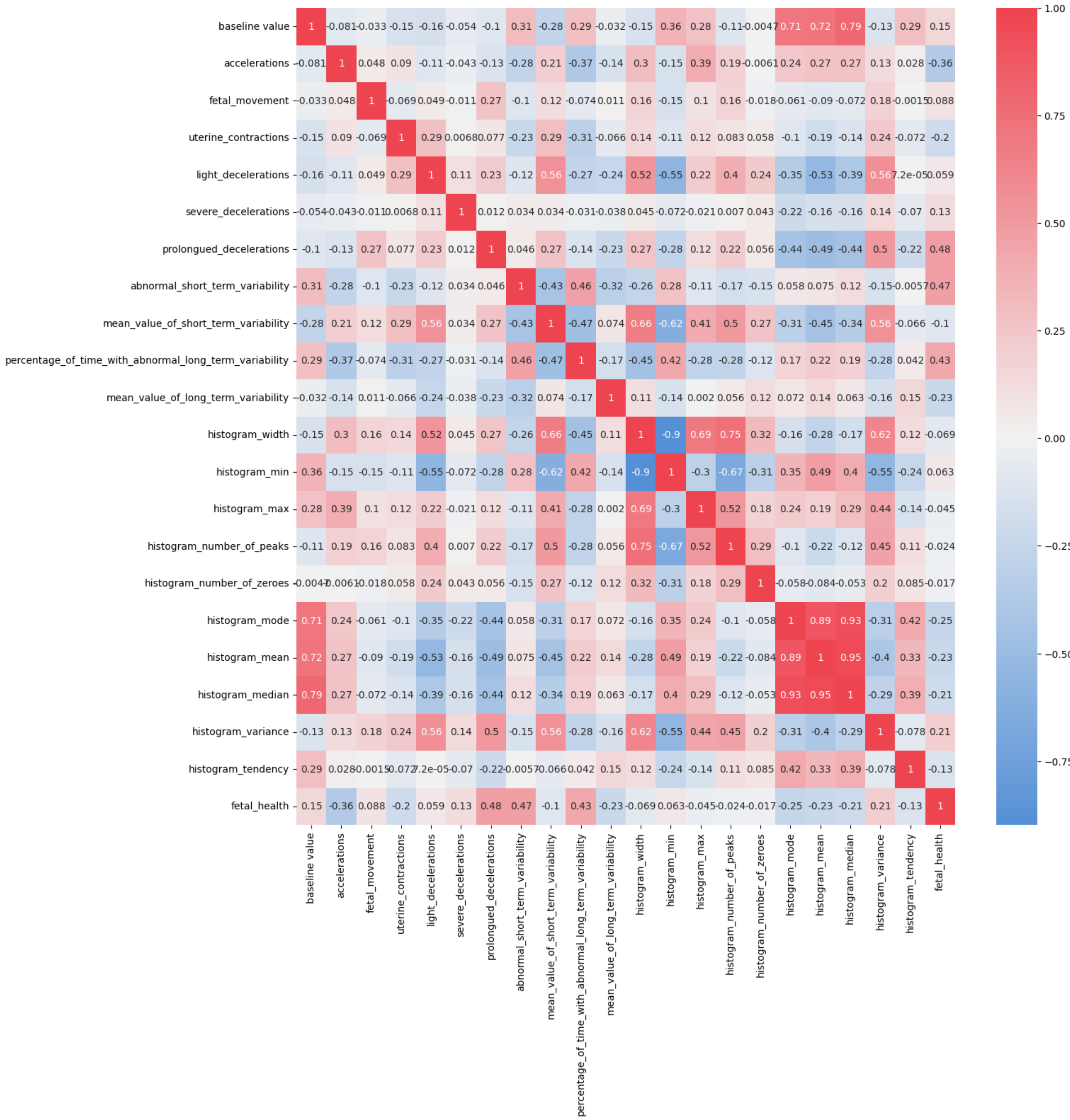
```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
baseline value	2126.0	133.303857	9.840844	106.0	126.000	133.000	140.000	160.000
accelerations	2126.0	0.003178	0.003866	0.0	0.000	0.002	0.006	0.019
fetal_movement	2126.0	0.009481	0.046666	0.0	0.000	0.000	0.003	0.481
uterine_contractions	2126.0	0.004366	0.002946	0.0	0.002	0.004	0.007	0.015
light_decelerations	2126.0	0.001889	0.002960	0.0	0.000	0.000	0.003	0.015
severe_decelerations	2126.0	0.000003	0.000057	0.0	0.000	0.000	0.000	0.001
prolongued_decelerations	2126.0	0.000159	0.000590	0.0	0.000	0.000	0.000	0.005
abnormal_short_term_variability	2126.0	46.990122	17.192814	12.0	32.000	49.000	61.000	87.000
mean_value_of_short_term_variability	2126.0	1.332785	0.883241	0.2	0.700	1.200	1.700	7.000
percentage_of_time_with_abnormal_long_term_variability	2126.0	9.846660	18.396880	0.0	0.000	0.000	11.000	91.000
mean_value_of_long_term_variability	2126.0	8.187629	5.628247	0.0	4.600	7.400	10.800	50.700
histogram_width	2126.0	70.445908	38.955693	3.0	37.000	67.500	100.000	180.000
histogram_min	2126.0	93.579492	29.560212	50.0	67.000	93.000	120.000	159.000
histogram_max	2126.0	164.025400	17.944183	122.0	152.000	162.000	174.000	238.000
histogram_number_of_peaks	2126.0	4.068203	2.949386	0.0	2.000	3.000	6.000	18.000
histogram_number_of_zeroes	2126.0	0.323612	0.706059	0.0	0.000	0.000	0.000	10.000
histogram_mode	2126.0	137.452023	16.381289	60.0	129.000	139.000	148.000	187.000
histogram_mean	2126.0	134.610536	15.593596	73.0	125.000	136.000	145.000	182.000
histogram_median	2126.0	138.090310	14.466589	77.0	129.000	139.000	148.000	186.000
histogram_variance	2126.0	18.808090	28.977636	0.0	2.000	7.000	24.000	269.000
histogram_tendency	2126.0	0.320320	0.610829	-1.0	0.000	0.000	1.000	1.000
fetal_health	2126.0	1.304327	0.614377	1.0	1.000	1.000	1.000	3.000

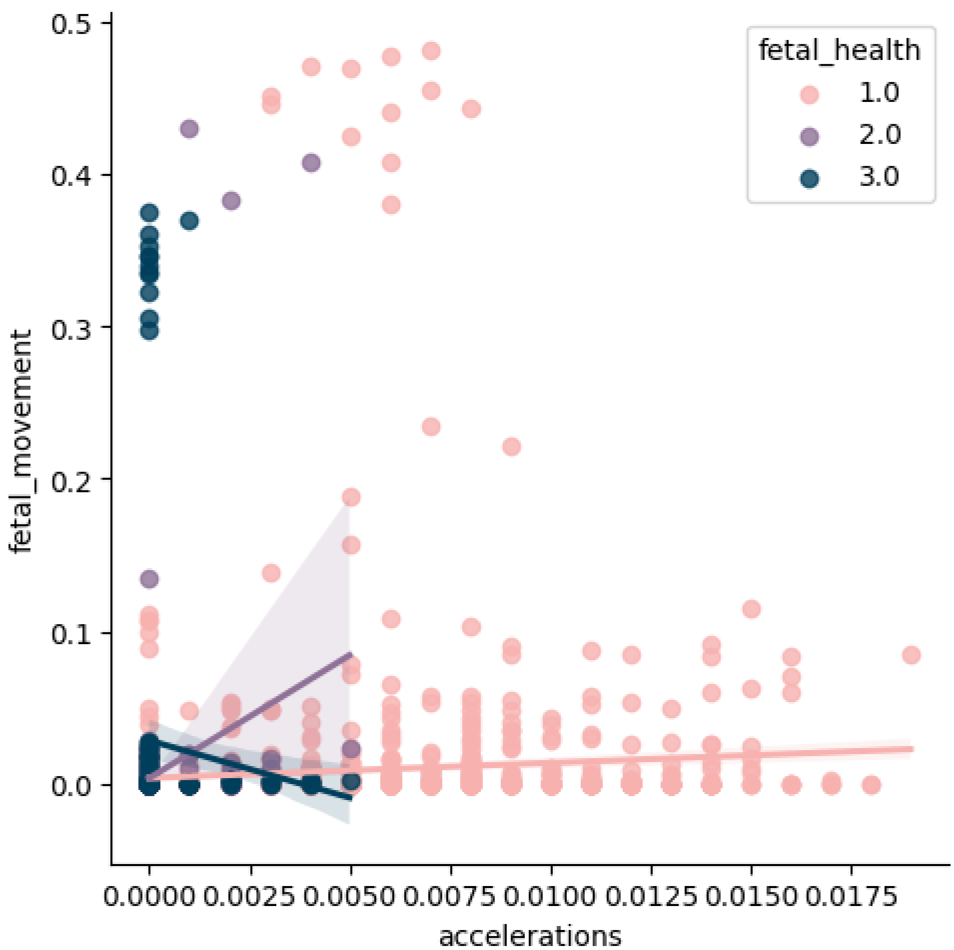
```
#first of all let us evaluate the target and find out if our data is imbalanced or not

colours=["#f7b2b0","#8f7198", "#003f5c"]
sns.countplot(data= data, x="fetal_health",palette=colours)
```



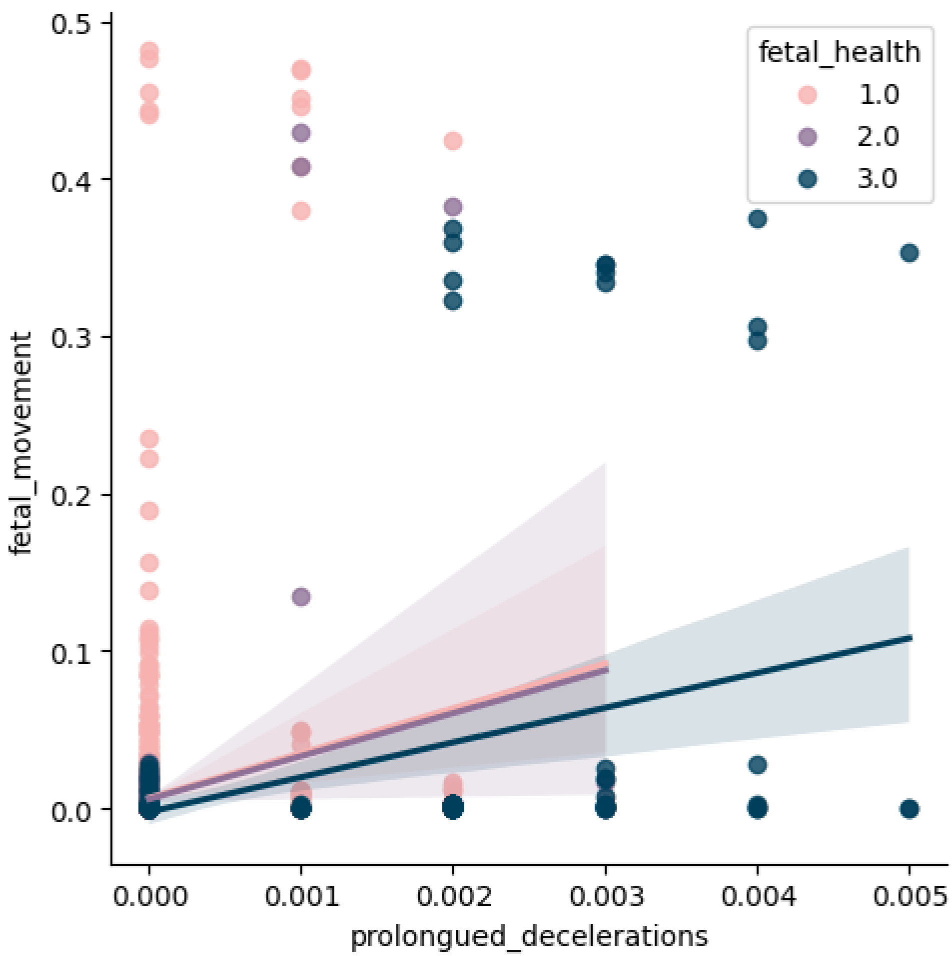

```
#correlation matrix
corrmat= data.corr()
plt.figure(figsize=(15,15))

cmap = sns.diverging_palette(250, 10, s=80, l=55, n=9, as_cmap=True)

sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
```



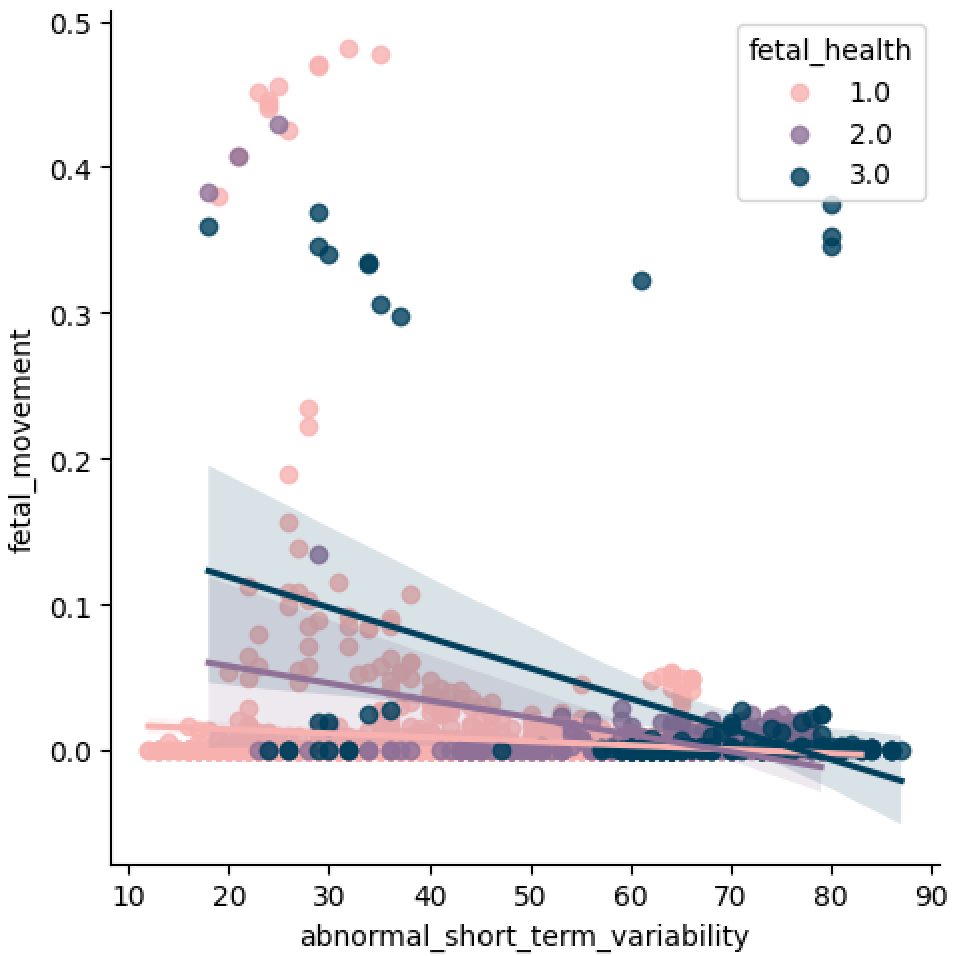

```
sns.lmplot(data
=data,x="accelerations",y="fetal_movement",palette=colours,
hue="fetal_health",legend_out=False)
plt.show()
```



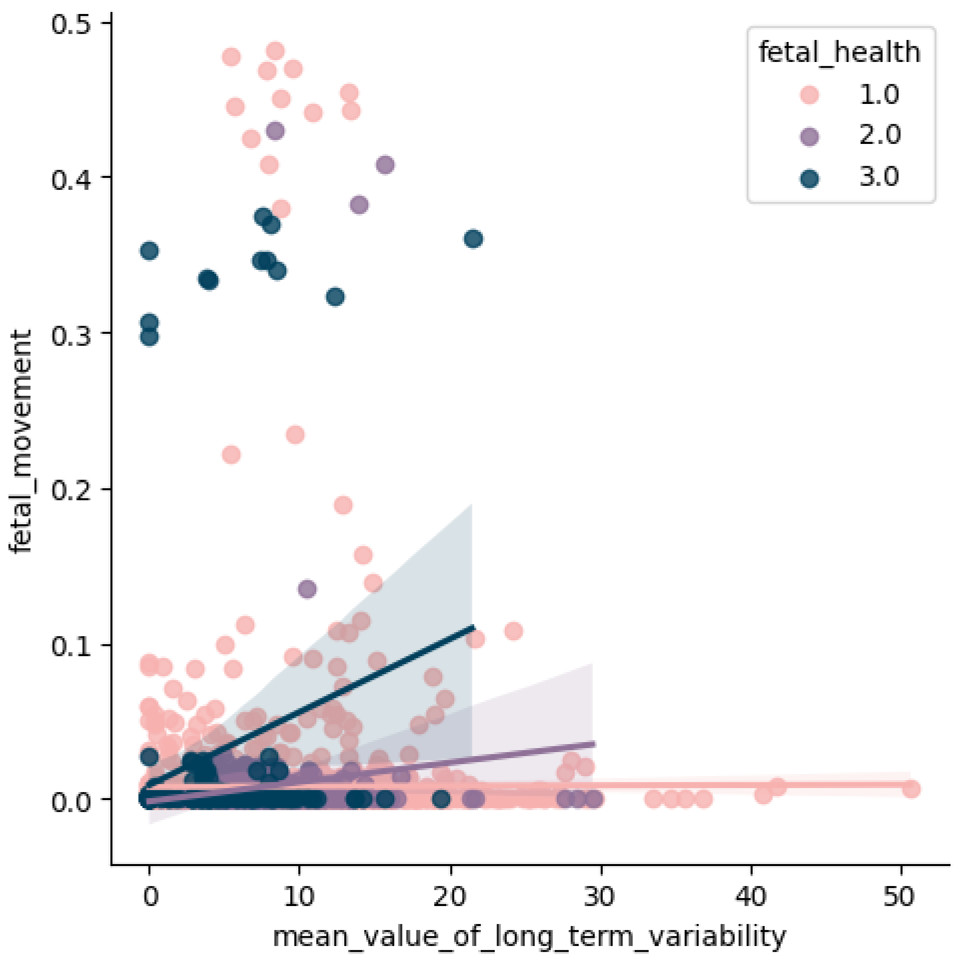

```
sns.lmplot(data
=data,x="prolongued_decelerations",y="fetal_movement",palette=colours,
hue="fetal_health",legend_out=False)
plt.show()
```



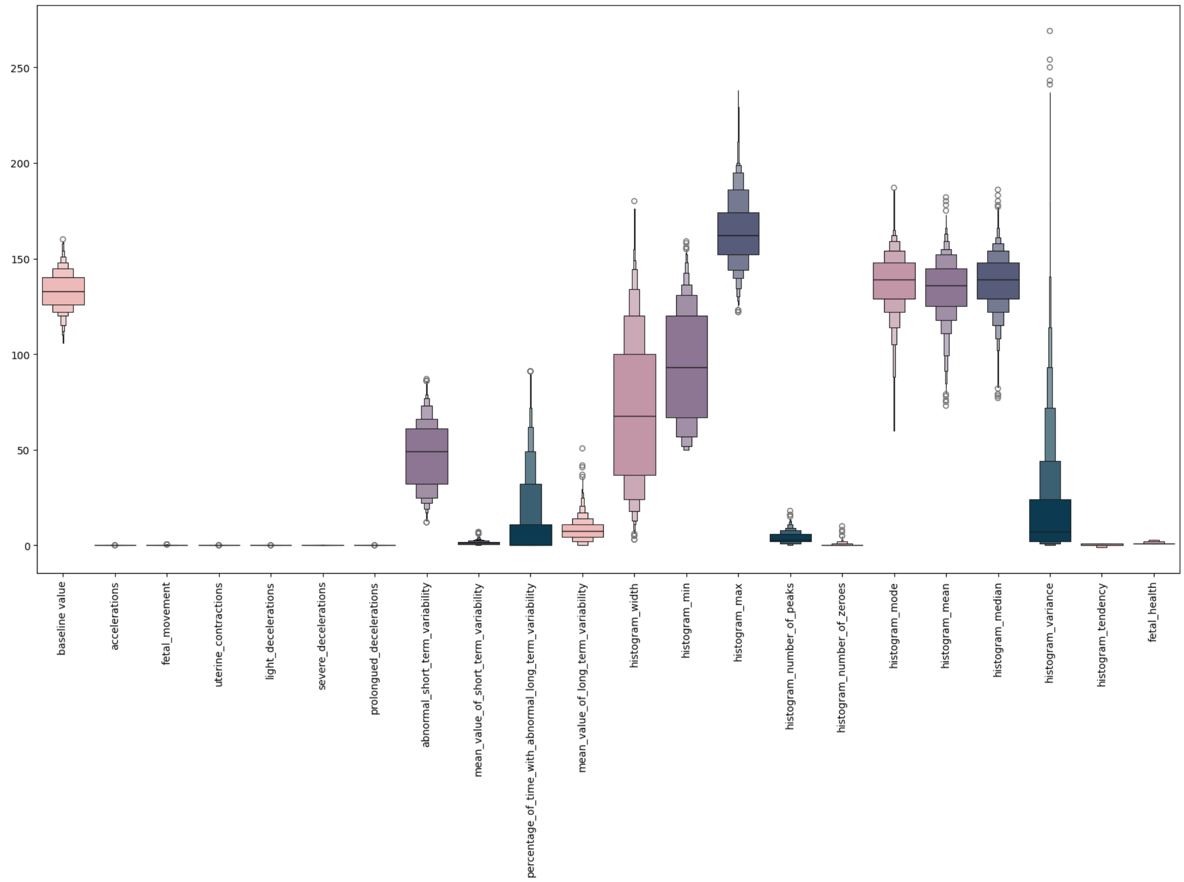

```
sns.lmplot(data
=data,x="abnormal_short_term_variability",y="fetal_movement",palette=c
olours, hue="fetal_health",legend_out=False)
plt.show()
```




```
sns.lmplot(data
=data,x="mean_value_of_long_term_variability",y="fetal_movement",palet
te=colours, hue="fetal_health",legend_out=False)
plt.show()
```



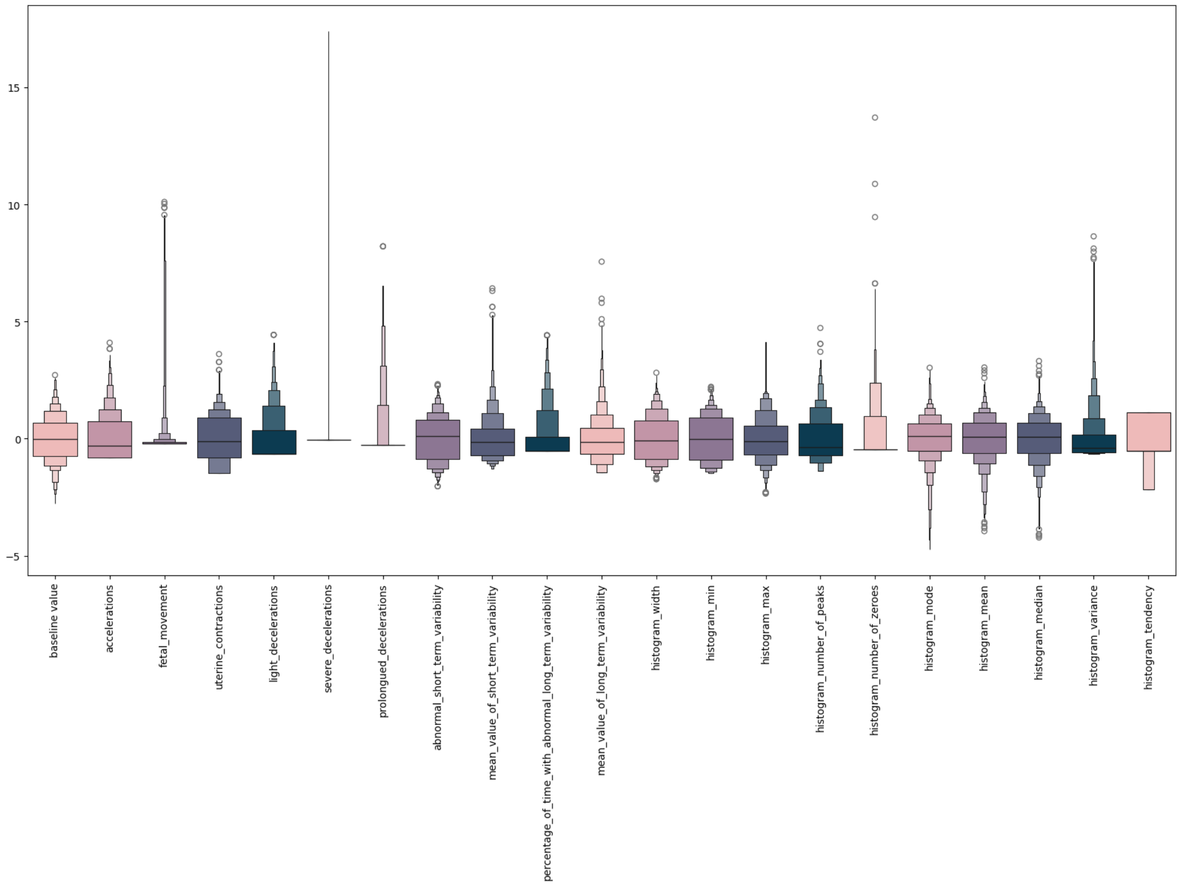

```
shades =["#f7b2b0","#c98ea6","#8f7198","#50587f", "#003f5c"]
plt.figure(figsize=(20,10))
sns.boxenplot(data = data,palette = shades)
plt.xticks(rotation=90)
plt.show()
```




```
#assigning values to features as X and target as y
X=data.drop(["fetal_health"],axis=1)
y=data["fetal_health"]

#Set up a standard scaler for the features
col_names = list(X.columns)
s_scaler = preprocessing.StandardScaler()
X_df= s_scaler.fit_transform(X)
X_df = pd.DataFrame(X_df, columns=col_names)
X_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
baseline value	2126.0	1.069490e-15	1.000235	-2.775197	-0.742373	-0.030884	0.680604	2.713428
accelerations	2126.0	-4.010589e-17	1.000235	-0.822388	-0.822388	-0.304881	0.730133	4.093929
fetal_movement	2126.0	-1.336863e-17	1.000235	-0.203210	-0.203210	-0.203210	-0.138908	10.106540
uterine_contractions	2126.0	-1.336863e-16	1.000235	-1.482465	-0.803434	-0.124404	0.894142	3.610264
light_decelerations	2126.0	-5.347452e-17	1.000235	-0.638438	-0.638438	-0.638438	0.375243	4.429965
severe_decelerations	2126.0	6.684315e-18	1.000235	-0.057476	-0.057476	-0.057476	-0.057476	17.398686
prolongued_decelerations	2126.0	1.336863e-17	1.000235	-0.268754	-0.268754	-0.268754	-0.268754	8.208570
abnormal_short_term_variability	2126.0	-7.352747e-17	1.000235	-2.035639	-0.872088	0.116930	0.815060	2.327675
mean_value_of_short_term_variability	2126.0	6.684315e-17	1.000235	-1.282833	-0.716603	-0.150373	0.415857	6.417893
percentage_of_time_with_abnormal_long_term_variability	2126.0	-5.347452e-17	1.000235	-0.535361	-0.535361	-0.535361	0.062707	4.412293
mean_value_of_long_term_variability	2126.0	2.406354e-16	1.000235	-1.455081	-0.637583	-0.139975	0.464263	7.555172
histogram_width	2126.0	-3.007942e-17	1.000235	-1.731757	-0.858765	-0.075640	0.758838	2.812936
histogram_min	2126.0	-4.679021e-17	1.000235	-1.474609	-0.899376	-0.019608	0.893996	2.213648
histogram_max	2126.0	-1.203177e-16	1.000235	-2.342558	-0.670314	-0.112899	0.555999	4.123453
histogram_number_of_peaks	2126.0	-1.671079e-16	1.000235	-1.379664	-0.701397	-0.362263	0.655137	4.724738
histogram_number_of_zeroes	2126.0	2.757280e-17	1.000235	-0.458444	-0.458444	-0.458444	-0.458444	13.708003
histogram_mode	2126.0	1.069490e-16	1.000235	-4.729191	-0.516077	0.094519	0.644055	3.025381
histogram_mean	2126.0	-6.684315e-16	1.000235	-3.951945	-0.616458	0.089126	0.666422	3.039749
histogram_median	2126.0	2.673726e-16	1.000235	-4.223849	-0.628514	0.062897	0.685166	3.312527
histogram_variance	2126.0	-5.347452e-17	1.000235	-0.649208	-0.580173	-0.407586	0.179212	8.635997
histogram_tendency	2126.0	-1.069490e-16	1.000235	-2.162031	-0.524526	-0.524526	1.112980	1.112980

```
#looking at the scaled features
plt.figure(figsize=(20,10))
sns.boxenplot(data = X_df,palette = shades)
plt.xticks(rotation=90)
plt.show()
```




```
#spliting test and training sets
X_train, X_test, y_train,y_test =
train_test_split(X_df,y,test_size=0.3,random_state=42)
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

# Define the logistic regression pipeline
pipeline_lr = Pipeline([('lr_classifier',
LogisticRegression(random_state=42))])

# Fit the logistic regression pipeline
pipeline_lr.fit(X_train, y_train)

# Perform cross-validation
```

```
cv_results_accuracy = cross_val_score(pipeline_lr, X_train, y_train,
cv=10)

# Print the cross-validation results
print("Logistic Regression: %f" % cv_results_accuracy.mean())
```

```
Logistic Regression: 0.897170
```

```
pred_lr = pipeline_lr.predict(X_test)
accuracy = accuracy_score(y_test, pred_lr)
print(accuracy)
```

```
0.8808777429467085
```

```
parameters_lr = {
    'lr_classifier__C': [0.1, 1, 10, 100],
    'lr_classifier__penalty': ['l1', 'l2'],
    'lr_classifier__solver': ['liblinear', 'saga']
}

# Perform GridSearchCV
CV_lr = GridSearchCV(estimator=pipeline_lr, param_grid=parameters_lr,
cv=5)
CV_lr.fit(X_train, y_train)

# Get the best parameters
best_params = CV_lr.best_params_

print("Best parameters for Logistic Regression:", best_params)
```

```
Best parameters for Logistic Regression: {'lr_classifier__C': 100, 'lr_classifier__penalty': 'l1', 'lr_classifier__solver': 'liblinear'}
```

```
# Create and fit the Logistic Regression model with the best
parameters
best_params_lr_extracted = {k.replace('lr_classifier__', ''): v for k,
v in best_params_lr.items()}
```

```python
LR_model = LogisticRegression(**best_params_lr_extracted,
random_state=42)
LR_model.fit(X_train, y_train)

# Test the model on the test set
predictions = LR_model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)

print("Accuracy of Logistic Regression model:", accuracy)
```

```
Accuracy of Logistic Regression model: 0.8824451410658307
```

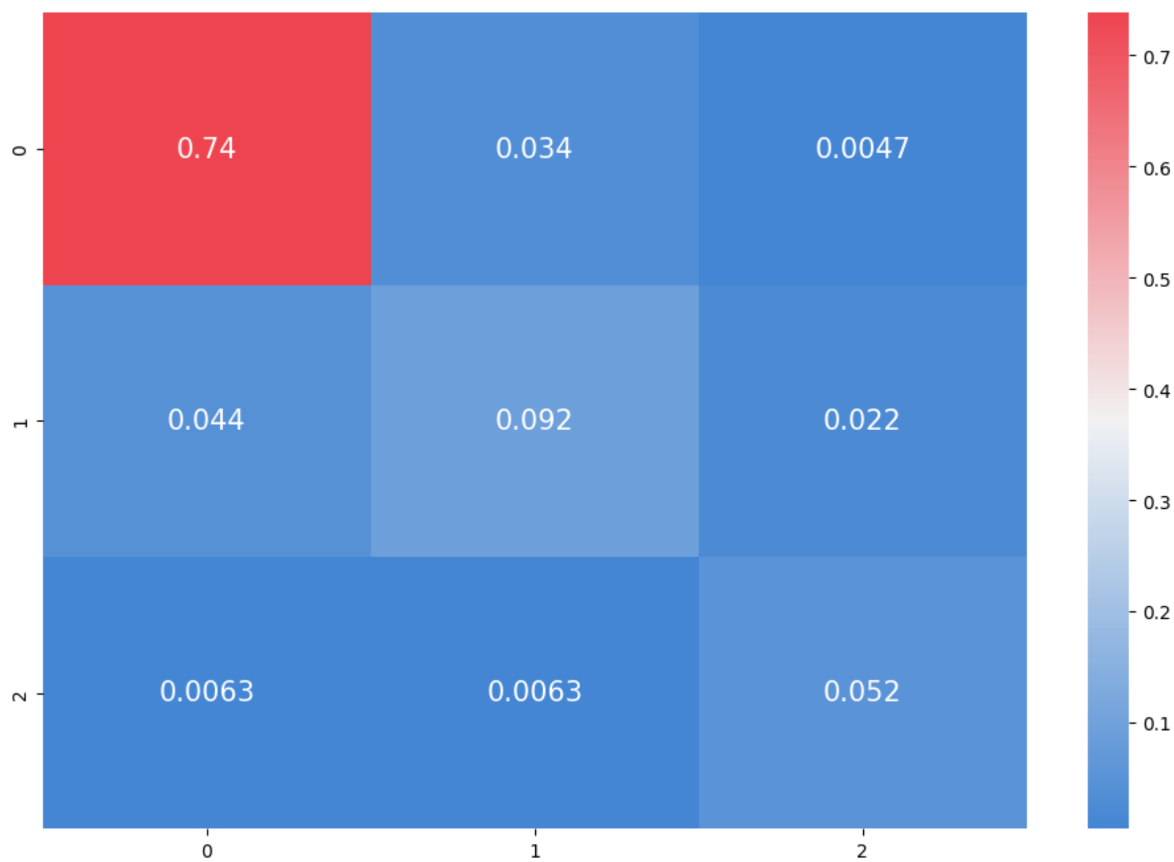
```python
acccuracy = accuracy_score(y_test, predictions)
recall = recall_score(y_test, predictions, average="weighted")
precision = precision_score(y_test, predictions, average="weighted")
f1_score = f1_score(y_test, predictions, average="micro")

print("********* Logistic Regression Results *********")
print("Accuracy    : ", acccuracy)
print("Recall      : ", recall)
print("Precision   : ", precision)
print("F1 Score    : ", f1_score)
```

```
********* Logistic Regression Results *********
Accuracy    :  0.8824451410658307
Recall      :  0.8824451410658307
Precision   :  0.880268354835032
F1 Score    :  0.8824451410658307
```

```python
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

         1.0       0.94      0.95      0.94       496
         2.0       0.69      0.58      0.63       101
         3.0       0.66      0.80      0.73        41

    accuracy                           0.88       638
   macro avg       0.76      0.78      0.77       638
weighted avg       0.88      0.88      0.88       638
```

```
# cofusion matrix
plt.subplots(figsize=(12,8))
cf_matrix = confusion_matrix(y_test, predictions)
sns.heatmap(cf_matrix/np.sum(cf_matrix), cmap=cmap,annot = True,
annot_kws = {'size':15})
```



Linear Regression:
Dataset: https://www.kaggle.com/code/karnikakapoor/fetal-health-classification

ALGORITHM:
Step 1: Create a sample dataset with multiple independent variables and one dependent variable (Y).
Step 2: The data is split into training and testing sets using the train_test_split function.
Step3: Different regression models are created and fitted to the training data.
Step4: Predictions are made on the test set.
Step5: The model is evaluated using metrics like Mean Absolute Error, Mean Squared Error,
and Root Mean Squared Error.
Step6: Finally, the coefficients and intercept of the regression equation are printed.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score,
mean_absolute_error
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
data = pd.read_csv('fetal_health.csv')

# Separate features and target
X = data.drop('fetal_health', axis=1)
y = data['fetal_health']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```python
# Create and train the model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Make predictions
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Evaluate the model on the test set
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
mae_test = mean_absolute_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)

# Evaluate the model on the training set
r2_train = r2_score(y_train, y_train_pred)

print(f"Train R-squared Score: {r2_train:.4f}")
print(f"Test Mean Squared Error: {mse_test:.4f}")
print(f"Test Root Mean Squared Error: {rmse_test:.4f}")
print(f"Test Mean Absolute Error: {mae_test:.4f}")
print(f"Test R-squared Score: {r2_test:.4f}")
```

```
Train R-squared Score: 0.6173
Test Mean Squared Error: 0.1566
Test Root Mean Squared Error: 0.3958
Test Mean Absolute Error: 0.2841
Test R-squared Score: 0.5400
```

Conclusion

I conducted an experiment using linear and logistic regression on a fetal health dataset. The linear regression model assessed feature impact on fetal health, with a train R-squared of 0.6173 (indicating how well the model fits the training data) and a test R-squared of 0.5400 (showing the model's predictive power on new data). The Test Mean Squared Error (0.1566) and Root Mean Squared Error (0.3958) measure prediction accuracy, while the Mean Absolute Error (0.2841) indicates average prediction error.

	The logistic regression model predicted fetal health classes with 88.24% accuracy, strong precision (94%) and recall (95%) for class 1.0, reflecting the model's effectiveness in identifying true positives and minimizing false positives. This experiment highlighted the importance of selecting appropriate regression techniques for healthcare analytics.