**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=β0 +β1 x+ϵ  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=β0 +β1 x1 +β2 x2 +…+βn xn +ϵ  where:  ● x1 ,x2 ,…,xn are the independent variables.  ● β1,β2,…,βn 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 (β0,β1,…,βn ) 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:  ● yi is the observed value.  ● 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 (β0 )**: Represents the expected mean value of y when all x variables are zero.  ● **Slope (β1,β2,…,βn )**: 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:  ● yˉ 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=β0 +β1 x+ϵ where y is the dependent variable, x is the independent variable, β0  is the y-intercept, β1 is the slope, and ϵ is the error term.  ○ **Multiple Linear Regression**: Extends simple linear regression by including  multiple independent variables:  y=β0+β1x1+β2x2+…+βnxn+ϵ  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=β0+β1x+β2x2+…+βnxn+ϵ  **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:    **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:    ○ **Lasso Regression:** Similar to ridge regression but uses an absolute value penalty,  which can shrink some coefficients to zero, effectively performing variable  Selection:    **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()    data.info()    data.describe().T    #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=colours, hue="fetal\_health",legend\_out=False)  plt.show()    sns.lmplot(data =data,x="mean\_value\_of\_long\_term\_variability",y="fetal\_movement",palette=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    #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())    pred\_lr = pipeline\_lr.predict(X\_test)  accuracy = accuracy\_score(y\_test, pred\_lr)  print(accuracy)    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)    # 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()}  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)    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)    print(classification\_report(y\_test, predictions))    # 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**](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)**  **# 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}")** |
| **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. |