Report

2.1 PreProcessing

Utilized UCI ‘`fetch_ucirepo`’ command to import the dataset from UCI database. This automatically stores it as a pandas DataFrame. **The dataset tries to predict whether a heart failure patient dies of a heart attack during a follow-up period window.**

Luckily the dataset did not contain null values. This was verified with the ‘`print(X.isna().any())`’ command which returned false for each column

The categorial values such as sex were already stored in binary form. I plotted the distribution of the binary categories with a count chart. Unfortunately, neither the dataset info page nor the research paper says which value corresponds with which class, so it is unknown whether 0 = male or female. There does however seem to be a noticeable discrepancy within each subcategory. Especially smoking and sex. Smoking makes sense since it is safe to assume there are fewer smokers than non-smokers, but the large discrepancy between sexes is surprising.

The distribution for the nonbinary classes was depicted using Histograms and Q-Q charts. The histogram essentially acts as a count chart, and the Q-Q chart shows the fit of the data’s quantiles in comparison to the normal distribution. Age, ejection faction, platelets, serum sodium, and time all fit relatively well on the distribution. On the other hand, creatinine phosphokinase and serum creatinine deviated significantly. This falls in line with how creatinine tends to fluctuate wildly in those with disease or health conditions. Specifically, muscle or health conditions. Since this data was from those with heart attacks, it makes sense that the variation is high.

The values were standardized using ‘`StandardScaler()`’ and ‘`scaler.fit_transform(X)`’

Or normalized using this: ‘`MinMaxScaler()`’ and ‘`scaler.fit_transform(X)`’

The correlation heat matrix generated with ‘`combined_df.corr()`’ and ‘`sns.heatmap()`’. This helped find key features i.e. the ones with the highest correlation to the target value (death): Age, ejection_fraction, serum_creatinine, serum_sodium, time)

Data was split into 80/20 using scikitlearns ‘`train_test_split()`’

2.2 Model Construction

The scikitlearn SGD Regressor and statsmodels OLS models were used as described in the instructions.

Optimal hyperparameters were found automatically with scikitlearns GridSearchCV. The tested parameters for the SGD Regressor model are as follows:

- max iterations : [10,100,500,1000,10000]
- learning rate : ['adaptive', 'invscaling', 'optimal', 'constant']
- loss : ['squared_error', 'squared_epsilon_insensitive', 'epsilon_insensitive', 'huber']
- epsilon : [1, .5, .3, .1, .01, .001, .0001, .00001]

the '`print(GS.best_params_)`' found this to be the best combination:
Max Iterations: 10; Learning rate: Invscaling; Loss: Squared Epsilon; Epsilon : 1 * 10^-5

2.3 Result Analysis

For the SGD Regressor model, the primary performance metrics used were Mean Squared Error (MSE) and R^2. The highest **MSE** I was able to achieve was **17.98%** after optimizing hyperparameters. This seemed like a reasonably low error value considering the lack of complexity in the model, but there is still room for improvement. The **R^2** value peaked at **26.03%**, indicating a relatively low tightness of fit on the testing data. This may be due to the large number of features (12) at play in this project, but it does make sense that the model generalizes poorly since the sample size is only 300. The **MAE** is **34.8%.** This is nearly a 20% difference from the MSE, indicating that heavy outliers are skewing the model's performance.

The OLS model produced similar results as follows – **MSE: 18.05%, R^2: 25.73%,** and **MAE: 34.75%**. Its important to note here that there are two R^2 values generated in my code. Since the .summary() function develops insight based on the training data fit, it differs slightly from the metrics calculated afterward via the scikitlearn library which uses testing data. In the .summary function my R^2 value was **43%**. This indicates that although the model fit the training data well, it generalizes poorly. Like the previous model, I believe this can be attributed to the low volume of data samples (300). At first I tested the model with all 12 features provided:

age, ejection fraction, serum creatinine, serum sodium, time, anaemia, creatinine phosphokinase, diabetes, high blood pressure, platelets, sex, smoking.

However, I used the correlations heatmap to determine which features contributed the most towards the model. I then removed features in the OSL table that had a P > |t| value greater than .05. I was happy to see the important features were the same across the two methods aside from one additional feature in the second.

Significant features after P > |t| threshold: age, ejection fraction, serum creatinine, time

This was very interesting to me as it signified diabetes, high blood pressure, and smoking did not contribute to death significantly as I initially expected. One interpretation of this could be that they contribute to heart attacks (potentially deadly in the moment), but not to death after those heart attacks.

The F-statistic was at **44.14%** which was nice to see. Since this shows how much better the model is than the null model, it depicts the variables and predictor made a significant impact.

The standard error was at an average of .023 across the features, which was significantly below the .05 alpha threshold. This indicates high prediction confidence.

The t values varied between -8 and 13, but in all honesty, I was not sure how to interpret this nor what could be the cause of it.