Machine Learning core lecture

Medical Expenses Prediction Challenge

Mohammad Shaique Solanki (7062950),
Shahriyar Haseeb Mansoor (7063015),
Shreyansh Tripathi (7056894)

Summer 2024

1 Introduction

This project, part of the Machine Learning core lecture for Summer 2024, aims to implement and evaluate predictive models to forecast individual healthcare utilization and total medical expenditure. Utilizing a dataset compiled from a U.S. patient survey, which includes demographic, personal, and health-related features, this study constructs a regression model to predict total medical expenditure and a classification model to determine healthcare utilization levels (low or high). The models will be developed using various machine learning techniques, evaluated on their accuracy and robustness, and refined through iterative testing with the provided datasets. The ultimate goal is to enhance resource allocation within healthcare services by enabling precise predictions of patient needs and costs, thereby offering a practical application of theoretical machine learning concepts learned throughout the course. The project submission will include detailed methodologies, empirical results, and additional analysis focused on data robustness and fairness, aiming to exceed the foundational knowledge provided by the lectures and contribute innovatively to the field of predictive healthcare analytics.

2 Dataset and Data Analysis

The dataset provided for the challenge was split into three parts, each in a Comma Separated Values (CSV) file. It included a total of 108 features (columns), along with two target variables: one for the regression task (TOT_MED_EXP) and another for the classification task (UTILIZATION). The pandas library was used to load the CSV files into the machine [4]. The details and sizes of each part are outlined below.

1. Training Data: Size - 15000x110 (108 features + Target + Label)
2. Test Public Data: Size - 4791x108
3. Test Private Data: Size - 5000x108

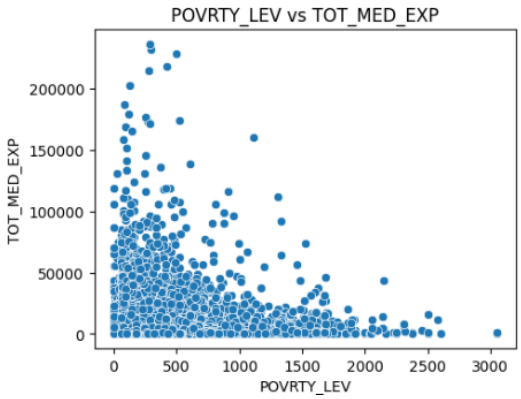
The analysis was conducted with two primary objectives:

1. To get an idea about the relationships between the target variable and the predictors.
2. To look at the areas our model needs to work on well to be considered fair.

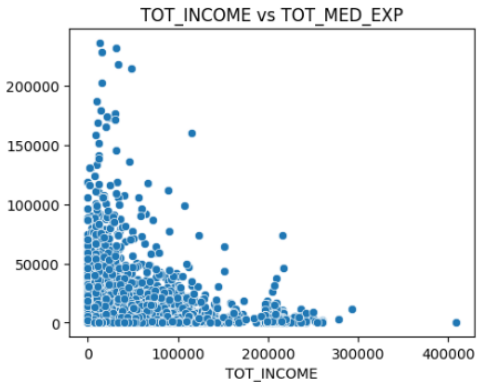
Initially, the target variable (TOT_MED_EXP) was plotted against various predictor variables. As depicted in Figure 1, the target variable exhibits significant variance, with most medical expenditures ranging from 0 to $10,000$, while some values exceed $225,000. Due to the broad range of the target variable, a log transformation was applied to facilitate a more nuanced understanding of the relationships between variables. This transformation aimed to normalize the data distribution, effectively reducing skewness. Fig 1 illustrates a comparison between the plots of several features against the log-transformed target variable and those against the untransformed target variable.

Fig 1 (c) and (d) show the **presence of heteroskedasticity** in the data as the variance of the target variable with respect to the features is different over the range. This is an important aspect used in the feature engineering of the regression model.

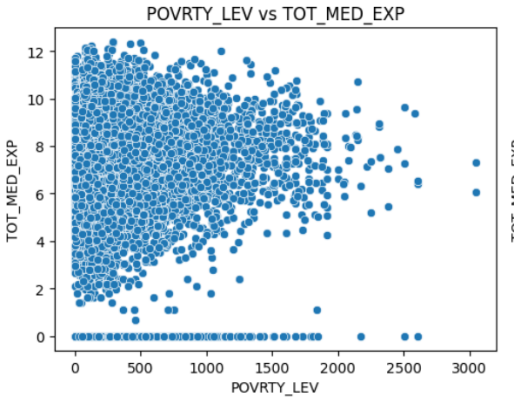
Fig 2 (a) and (b) show the box plots of the target variable for sex and race respectively. It can be seen from the figure that both the median and the 25 percentile expenditure in the non-white community is lower than in



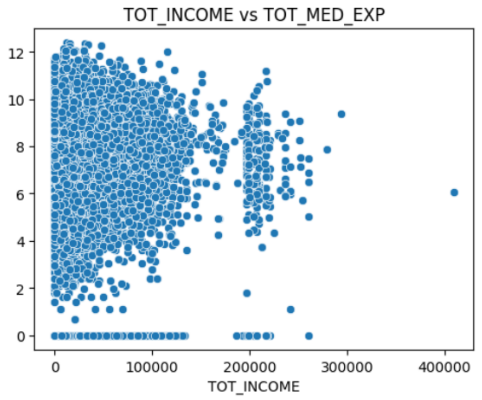
(a) Poverty Leverage vs Total Expenditure



(b) Total Income vs Total Expenditure

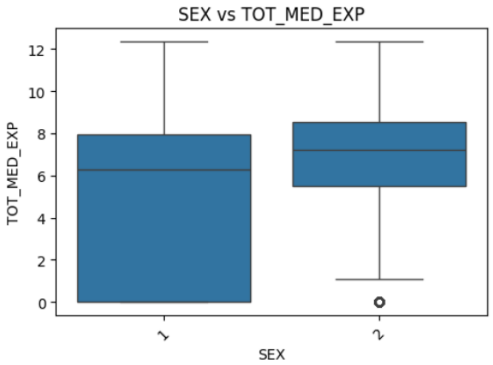


(c) Poverty Leverage vs log-transformed Total Expenditure

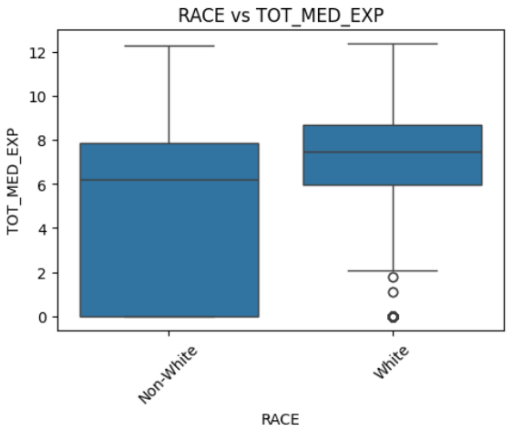


(d) Total Income vs log-transformed Total Expenditure

Figure 1: Scatter Plot of the features with target variable



(a) Sex vs Total Expenditure



(b) Race vs Total Expenditure

Figure 2: Box Plot of the log-transformed Total Expenditure with sex and race

the white community. The difference is similar when comparing the sexes. This observation may point towards structural inequality between sexes and races and thus to make modelling fairer, these classes may be treated separately for an accurate representation of these inequalities.

3 Experiments and Results

For this project, the models were compiled using the Sklearn library in python. Sklearn provides robust APIs for required models with very good flexibility to tune the parameters. [1].

3.1 Metrics Used:

Regression Task:

1. Root Mean Squared Error(RMSE): measures the square root of the average of squared differences between predicted and actual values. It gives higher weight to large errors, making it sensitive to outliers.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

2. Mean Absolute Error(MAE): provides a more straightforward interpretation of the average magnitude of errors, as it doesn't square the differences, making it less sensitive to outliers.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of data points.

Classification Task:

The evaluation metric used for the classification task is the Macro-Averaged F1 score.

The macro-averaged F1 score is calculated by taking the average of the F1 scores of all classes. The F1 score for each class is given by:

$$F1_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

where $\text{Precision}_i = \frac{TP_i}{TP_i+FP_i}$ and $\text{Recall}_i = \frac{TP_i}{TP_i+FN_i}$, with TP_i, FP_i, and FN_i representing the true positives, false positives, and false negatives for class i, respectively.

The macro-averaged F1 score is then:

$$F1_{\text{macro}} = \frac{1}{N}\sum_{i=1}^{N} F1_i$$

where N is the number of classes.

In this challenge, the value of $N = 2$.

3.2 Regression Task

3.2.1 Preprocessing

1. **Data Cleaning**: We eliminated the values -7, -8, and -9 from the training dataset. This has been due to the following reasons:
   (a) The test set did not have any of these values.
   (b) The label-encoded features should have values above 0.
2. **Feature Selection**: We have used two methods for the task of feature selection, namely SelectKBest and Recursive Feature Elimination with Cross-Validation(RFECV).
   (a) SelectKBest: selects the top K features based on a statistical test or scoring function. We have used the f_regression scoring function that calculates the F-value between label/feature for regression tasks. After scoring, the SelectKBest method selects the top K features with the highest scores. Here we have selected 88 features for this technique. Since K is a hyperparameter, we used the cross-validation technique on 5 folds to conclude the K value as 88 features.
   (b) RFECV: recursively removes less important features and evaluates model performance using cross-validation to determine the optimal number of features. RFECV automatically finds the optimal number of features through cross-validation. We got 80 features by using this method.
3. **One hot encoding**: The 'RACE' feature was one-hot encoded since it is a categorical feature. One hot encoding allows categorical data to be represented numerically without implying any ordinal relationship. An ordinal relationship implies a natural order or ranking among the categories.

3.2.2 Model Selection

We have used 4 models namely,

1. Linear Regression
2. XGboost [2]
3. Gradient Boosting [3]
4. Neural Networks [5]

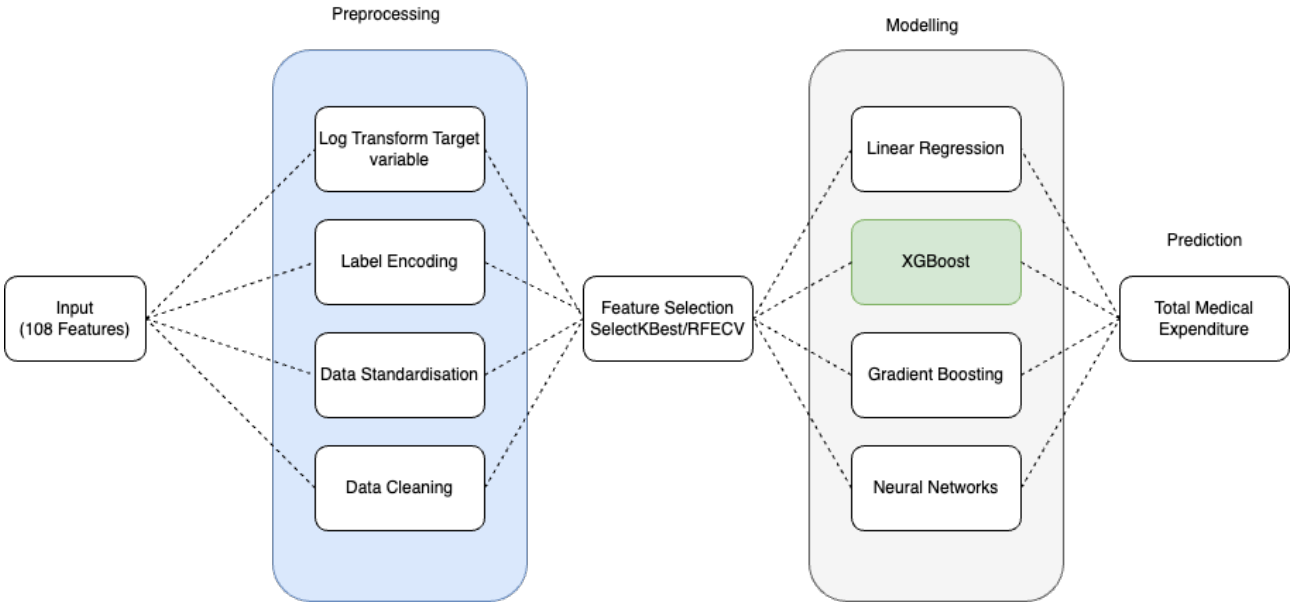


Figure 3: Regression Pipeline Used. The pipeline includes preprocessing, feature selection and Modelling steps

3.2.3 Training Procedure

After the above steps, we split our dataset into train, validation, and test splits. The train-test split is of the 70:30 ratio and we ran the models with a 5 fold cross validation. The final hyperparameters for best performing xgboost model are:

n_estimators: 600 colsample_bytree: 1.0
min_child_weight: 50 max_depth: 3
gamma: 1.5 learning_rate: 0.05
subsample: 0.8

3.2.4 Results

Table 1: Mean Absolute Error and Root Mean Squared Error Metrics on different ML Regressors before and after feature selection. The best-performing regressor is marked in green (FS: Feature Selection)

Models	MAE		RMSE	
	Before FS	After FS	Before FS	After FS
Linear Regression	4790.87	4565.89	14787.28	13758.46
XgBoost	3245.76	3109.78	10721.38	10704.79
Gradient Boosting	3465.23	3429.92	11237.19	10932.48
Neural Networks	3749.29	3599.26	10786.16	10735.23

3.3 Classification Task

The objective was to predict the High/Low usage of the patient's healthcare usage based on 108 features as mentioned in subsection 4.1.1. (Column Name: UTILIZATION)

The dataset was split into train and validation sets with a split ratio 0.3. Since the dataset concerning labels is not balanced, stratification was done to maintain the proportionality of the labels in both train and validation sets.

The step-wise classification pipeline is shown in Fig. 4

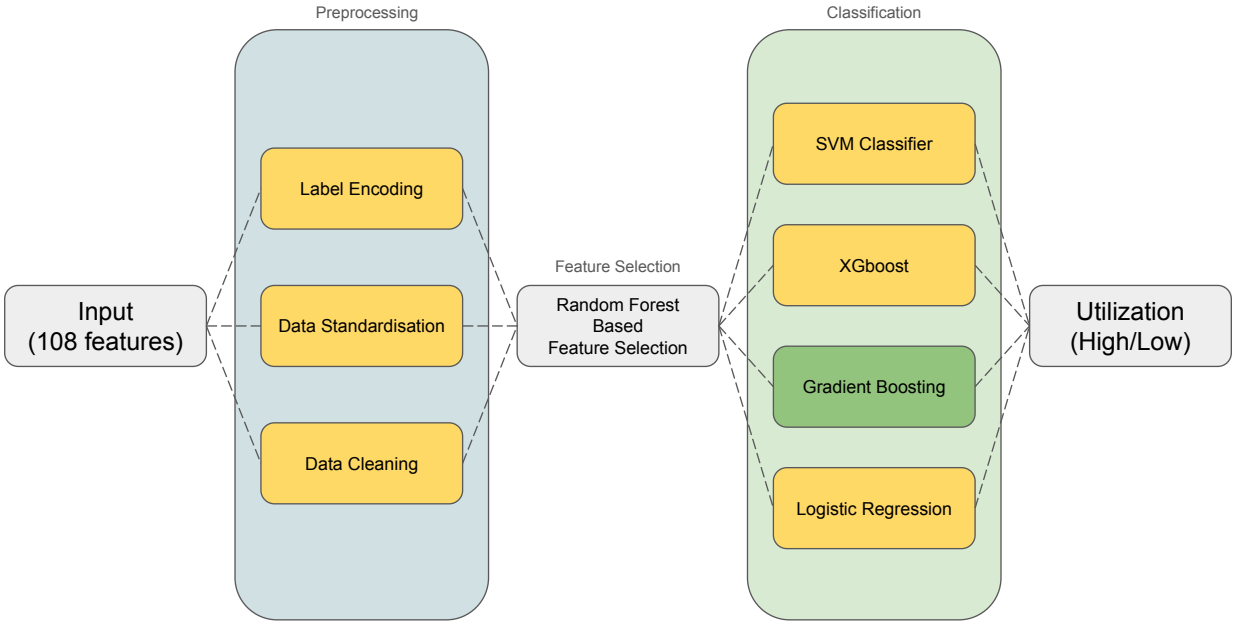


Figure 4: Classification Pipeline

3.3.1 Preprocessing

- **Encoding**: All the categorical features in the dataset, such as RACE, SEX, PREGNT,.. etc. are encoded using the Label Encoding technique. Label Encoding assigns a unique number to each category.
- **Standardisation**: All the features were standardised to have zero mean and unit variance. In simpler terms, the range of values each feature could take was [0,1].
- **Data Cleaning**: The data was scanned to eliminate missing values(if any) to be replaced by 0 for numerical features, such as weight, age, etc, and a null value in case of categorical data features such as RACE, SEX etc.

3.3.2 Feature Selection

Random Forest algorithm is often used as a feature selection step in many machine learning tasks [6]. Key features that enable Random Forest algorithm to be a good feature selector are:

1. **Intrinsic Feature Ranking**: Random Forest provides a built-in method to evaluate the importance of features.
2. **Handles High Dimensionality**: Effective even when the number of features is much larger than the number of samples.
3. **Non-Linearity**: Can capture complex interactions between features without requiring explicit specification of interactions.

The feature importance vector given by the random forest algorithm was thresholded at the median and all the features with a higher importance than the threshold were selected.

Multiple ML models were then trained on these selected features and their performance was compared. The model that gave the best Macro Average F1 score both, before and after feature selection was then used to predict labels on the test data. This is further discussed in the subsection 5.2.3

3.3.3 Classification:

Four Machine Learning models as listed below were trained on the dataset for both, pre and post-feature selection.

1. Random Forest
2. Logistic Regression
3. Gradient Boosting
4. Support Vector Machine(SVM)
5. XGboost

The hyperparameters of each model listed in the table above were tuned using 5-fold cross-validation with train and validation set split of 0.3.

It has to be noted here that, the quantity of the data points of each label is not equal, i.e. the dataset is imbalanced and hence there is a risk of classifier bias, to enumerate, there are 11788 data points that are labelled as "LOW" whereas only 3212 are labelled as "High".

To solve this problem, a stratification technique was used while splitting the train and validation set so that the proportionality is maintained.

The table 2 summarises the effect of feature selection on the above-listed models.

Fig 5 summarizes the ROC curve(s) and confusion matrices on the validation set of each classifier.

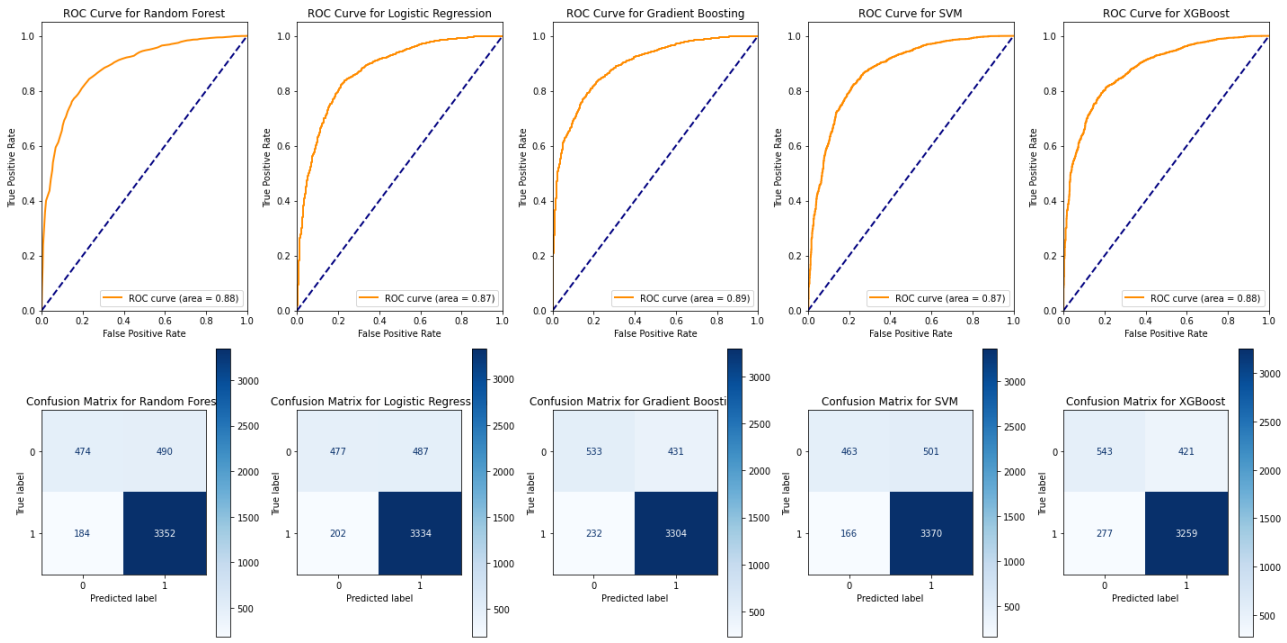


Figure 5: ROC curves and confusion matrices on the validation set of each classifier after feature selection

Table 2: F1 score (macro-avg) of different ML Classifiers before and after Random Forest-based feature selection. The best-performing classifier is marked in green.

Model	F1 score (macro-avg)	
	Before Feature Selection	After Feature Selection
Random Forest	0.7466	0.7556
Logistic Regression	0.7435	0.8478
Gradient Boosting	0.7627	0.7636
SVM	0.7355	0.7465
XGBoost	0.756	0.7494

References

[1] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[3] A. Natekin and A. Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[4] T. pandas development team. pandas-dev/pandas: Pandas, Feb. 2020.

[5] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

[6] N. Pudjihartono, T. Fadason, A. W. Kempa-Liehr, and J. M. O'Sullivan. A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, 2:927312, 2022.