How to Cite:
Vijaya, K. S., Jena, G., Vannika, T., Akhil, T. M., Balapraveen, U., & Chaitanya, T. A. S. (2022). Patient expenditure prediction using deep learning framework. *International Journal of Health Sciences*, *6*(S4), 4528–4540. https://doi.org/10.53730/ijhs.v6nS4.9109

Patient expenditure prediction using deep learning framework

Ms. K Sasi Vijaya
Asst Prof CSE, BVCEC
Corresponding author email: sasivijaya.k@gmail.com

Dr Gunamani Jena
Prof CSE, BVCEC
drgjena@gmail.com

Tirumani Vannika
CSE, BVCEC

Tula Manu Akhil
CSE, BVCEC

Undru Balapraveen
CSE, BVCEC

Tanneedi Anjan Sai Chaitanya
CSE, BVCEC

***Abstract*---**Measurement of patient expenditure in healthcare is a critical task that has a variety of applications, including provider profiling, accountable care management, and capitated medical payment adjustment. Currently available methods rely on manually built features and linear regression-based models, both of which need a significant amount of medical domain expertise. and have low prediction accuracy. This study develops a multi-view deep learning system for forecasting future healthcare costs at the individual level based on prior claims data. Our multi-view technique efficiently models heterogeneous data such patient demographics, medical codes, medication usages, and facility usage. To execute spending forecasting tasks, we employed a real-world paediatric dataset with approximately 450,000 patients. According to the empirical data, our proposed technique beats all baselines for predicting medical cost. In the sphere of healthcare, these insights help to improve preventative and responsible care.

***Keywords*---**deep learning, electronic health records, spending prediction, machine learning, analyse administrative claims data.

Introduction

As a result of escalating healthcare expenses, healthcare providers and organisations are facing a significant problem. The United States' national health expenditure (NHE) grew by 4.6 According to the Centers for Medicare & Medicaid Services (CMS), Medicare spending increased by 17.7% to $3.6 trillion in 2018 (or $11,172 per person), accounting for 17.7% of GDP (GDP). Medicare expenditure jumped by 6.4 percent to $750.2 billion, while Medicaid spending rose by 3.0 percent to $597.4 billion. 1 If medical costs continue to rise at their present pace, the healthcare system will become unsustainable.[1].

It is vital to keep healthcare expenditures from rising out of control and to reduce individual medical bills. Claims data is a sort of Electronic Health Records (EHR) that includes demographics, diagnoses, treatments, medications, and facility information. One of the most thorough sources for analysing individuals' health concerns is claims data from the National Health Expend Data System. The increasing volume of claims data presents a fresh and promising way to tackling healthcare cost problems. Data-driven models may be constructed using historical claims as a starting point to provide critical insights into expenditure trends. For example, an accurate medical cost projection model at the individual level can aid in the identification of persons who are at high medical risk and the provision of better treatment.

The majority of existing solutions for forecasting patient expenditures [2], [3] involve handcrafted characteristics and linear regression-based models. The Diagnostic Cost Groups (DCG) [4] technique, for example, uses linear regression to forecast healthcare costs based on diagnostic classifications created by topic specialists manually. Based on the aggregate medical codes and custom cost components, Bertsimas et al. [5] constructed a Classification And Regression Tree (CART). These models aid in the forecasting of healthcare costs.

However, they are limited by the following factors: 1) To organise high-dimensional medical codes into semantically relevant categories, they depend heavily on domain knowledge. 2) They overlook a wealth of data in claims data, such as facility use, temporal data, and medical code connections. 3). The predictive ability of the linear regression-based model is limited., resulting in sub-optimal model performance. The purpose of this research is to fill in the gaps in the literature. Our research has been affected by a number of observations. To begin with, claim data includes a wide range of information. As indicated in Figure 1, various domains, Medical codes, facility usages, and demographics, for example, may include a variety of attributes and data formats. These domains are useful for budgeting, but modelling them using general-purpose machine learning or deep learning models is difficult.

A multi-view deep learning system for learning and predicting medical expenditures and learning efficient and interpretable patient representations. To

utilise heterogeneous information in claims data from many perspectives, we use a feed forward neural network, an attention-based bidirectional recurrent neural network, and a hierarchical attention network.

The purpose of this study was to develop a model that could accurately forecast expenditures based on prior claims. Prior study has indicated that healthcare expenditure is episodic and inconsistent, whereas other research has discovered that healthcare spending is strongly linked to time. The usefulness of projecting expenses by modelling past claims is related to the degree of unpredictability in the healthcare utilisation pattern. We looked at the temporal connection of yearly spending at the person level using our paediatric dataset.

This study develops a multi-view deep learning system for forecasting future healthcare costs at the individual level based on prior claims data. Our multi-view technique efficiently models heterogeneous data such patient demographics, medical codes, medication usages, and facility usage. To execute spending forecasting tasks, we employed a real-world paediatric dataset with approximately 450,000 patients.

Formulation

In this study, we apply Random Forest, Extreme Gradient Boosting, Decision Tree, Ada Boost, CNN, and LSTM. The LSTM solves issues by learning new information and forgetting old information via assigned gates. A cell, an output gate, and a forget gate are the three components of a standard LSTM unit. The cell's main function is to recognise values across random time intervals, while the gates control information flow into and out of the cell.

The random forest approach relies on a variety of arbitrary assumptions, such as collecting training data at random, splitting nodes at random, and only considering a fraction of all attributes when dividing each node in a basic decision tree. Random forest trees learn from a random sample of data points during training. In a random forest model, there are a plethora of trees. By averaging the predicted outcomes of trees, the model creates a forest.

Three random notions are used in the method: randomly choosing training data when creating trees, randomly selecting specific subsets of variables when dividing nodes, and only considering a fraction of all variables when splitting each basic decision tree's nodes. For each basic tree in a random forest, the dataset is drawn from a random sample. For in-built cross-validation, regularisation (to minimise overfitting), rapid handling of missing data, catch awareness, tree trimming, and parallelized tree construction, Extreme Gradient Boosting is used.
A Choice Tree's purpose is to build a model that can predict a target value based on basic decision rules learned from input qualities. This approach offers a number of advantages, including being easy to perceive and grasp, as well as the ability to handle problems with several outputs.

Boost Ada

Boosting techniques are a collection of algorithms that aid in the development of weak learners. Boosting is a method of teaching poor learners consecutively in order to modify their previous predictions.

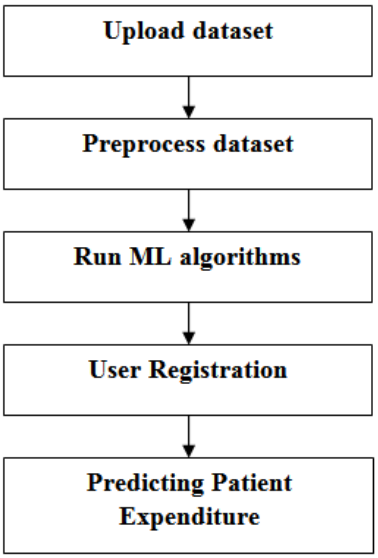
CNN

The term "neural networks" refers to a notion. In a typical Neural Network, there are three types of layers. Input Layers (Layers of Input): It's the layer where we input our model data. The number of neurons in this layer is equal to the total number of attributes in our data (number of pixels in the case of an image). Layer that isn't visible: The input layer sends the data to the hidden layer. There might be a lot of hidden layers depending on our model and the amount of data we have. The hidden layer's output is then sent into a logistic function like sigmoid or soft max, which converts each class's output into a probability score for that class.

Voting classifier

There are many different types of voting classifiers, and they all use machine learning to predict the outcome of a vote. Each estimator's output may be aggregated using a voting criteria. After we've trained all of the algorithms, we need to double-check their accuracy. The algorithm with the best accuracy will be used to anticipate spending. Following the discovery of the best algorithm, we enter the user's data, and the algorithm predicts the optimum medical expenditure for him/her based on the user's data.

Algorithm and Process Design



Flow chart 1: predicting patient expenditure

Algorithm-1

1. Upload dataset: We will upload the dataset using this module.
2. Clean dataset: Using this module, remove null values from the dataset.
3. Run machine learning algorithms: this module is used to partition train and test data in order to construct a model for prediction utilising all machine learning methods.
4. User Registration: This module is used to register new users on the site.
5. Predicting Patient Spending: We will use this module to submit patient information after patient expenditure has been calculated.

Data Gathering

Partner for Kids gathered administrative claims data from the Medicaid programme (PFK). PFK is one of the nation's biggest nonprofit health care organisations, providing integrated services to children throughout south-central and southeast Ohio. From January 2013 to December 2014, we collected more than 8,500,000 medical records from 450,000 individuals. Enrollees must be eligible to participate in the studies from January 2013 to December 2014. Continuous eligibility enforcement is used to keep patients who have just been enrolled for a brief time yet have significantly changeable profiles compared to the general population out of the study.

Metrics for Evaluation

Area, F1-Score, and Accuracy of Receiver Operation The Receiver Operating Characteristics Area Under the Curve (ROC-AUC) measurements are what we use to gauge how well our models are working. In order to acquire an F1-score and Accuracy, the FPR=False Positive Rate must be calculated.
True Positive Rate is referred to as TPR.

Accuracy

Accurately recall the F1-score
Based on the following, values are computed: "True positive" (TP) indicates that the number of occurrences was correctly determined.
False negative (FN): the number of occurrences that were mistakenly anticipated and unnecessary.
FP = the number of occurrences that were incorrectly predicted.
"True negative" (TN) refers to the amount of events that have been correctly anticipated but are not essential.
Machine learning accuracy is measured by the False Positive Rate (FPR). In layman's terms,
FPR=FP/(FP+TN)
True Positive Rate (TPR): As a synonym for recall, TPR is defined as FP/(FP+TN).
Accuracy: The most essential performance metric is the ratio of accurately predicted observations to total observations, which is simple to calculate.
Accuracy=(TN+TP)/(TP+FP+TN+FN)
Recall: This is the ratio that properly anticipates positive observations in the original data from all observations.

TP/(TP+FN) = Recall
Precision is utilised to compute the values that have been accurately identified. This implies subtracting the total number of accurately predicted positive softwares from the total number of softwares anticipated positive. Precision is defined as TP/(TP + FP).
The F1-score is a method of combining the model's accuracy and recall, and it is defined as the mean of the model's precision and recall. It's also known as the F-score. F1 Score = 2(Precision Recall/Precision + Recall) is the formula.
The area under the ROC-AUC curve from prediction scores is used to calculate ROC-AUC, which is another useful statistic for classification issues.

Models used in this work

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor(max_depth=2, random_state=0)
RF.fit(X_train, y_train)
predictions = RF.predict(X_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble
l change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
print(RF.score(X_test, y_test))
```

```
0.47468041055548016
```

```
from sklearn.metrics import mean_squared_error
RF = mean_squared_error(y_test, predictions)
RF
```

```
21312240.455974203
```

Voting Regressor

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import VotingRegressor
r1 = LinearRegression()
r2 = RandomForestRegressor(random_state=1)
er = VotingRegressor([('lr', r1), ('rf', r2)])
er.fit(X_train, y_train)
predictions = er.predict(X_test)
```

```
In [40]: print(er.score(X_test, y_test))
```

```
         0.689932961863369
```

```
In [41]: from sklearn.metrics import mean_squared_error
         vot = mean_squared_error(y_test, predictions)
         vot
```

```
Out[41]: 12579434.323451042
```

Linear Regression

```
from sklearn.linear_model import LinearRegression
r1 = LinearRegression()
r1.fit(X_train, y_train)
predictions = r1.predict(X_test)
print(r1.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
lr = mean_squared_error(y_test, predictions)
lr
```

```
0.6218888805277105

15339985.904184693
```

Decision Tree Regressor

In [43]:
```
from sklearn.tree import DecisionTreeRegressor
r1 = DecisionTreeRegressor(random_state=0)
r1.fit(X_train, y_train)
predictions = r1.predict(X_test)
print(r1.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
dt = mean_squared_error(y_test, predictions)
dt
```

```
0.4649130674734097
```

Out[43]:
```
21708502.024291497
```

FNN - MLP

In [44]:
```
from sklearn.neural_network import MLPRegressor
regr = MLPRegressor(random_state=1, max_iter=500).fit(X_train, y_train)
predictions = regr.predict(X_test)
print(regr.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
fnn = mean_squared_error(y_test, predictions)
fnn
```

```
0.21109183621194216

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neural_network\multi
Optimizer: Maximum iterations (500) reached and the optimization hasn't
  % self.max_iter, ConvergenceWarning)
```

Out[44]:
```
32006041.316888433
```

Gradient Boosting Regressor

```
from sklearn.ensemble import GradientBoostingRegressor
reg = GradientBoostingRegressor(random_state=0)
reg.fit(X_train,y_train)
predictions = reg.predict(X_test)
print(reg.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
gnb = mean_squared_error(y_test, predictions)
gnb
```

```
0.6222334538833285

15326006.546937766
```

LAsso Model

In [46]:
```
from sklearn import linear_model
clf = linear_model.Lasso(alpha=0.1)
clf.fit(X_train,y_train)
predictions = clf.predict(X_test)
print(clf.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
lasso = mean_squared_error(y_test, predictions)
lasso
```

```
0.6218752529420433
```

Out[46]:
```
15340538.775976282
```

Ridge

```
from sklearn.linear_model import Ridge
clf1 = Ridge(alpha=1.0)
clf1.fit(X_train,y_train)
predictions = clf1.predict(X_test)
print(clf1.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
rid = mean_squared_error(y_test, predictions)
rid
```

```
0.6199842380999128
```

```
15417257.337079095
```

AdaBoost Regressor

```
from sklearn.ensemble import AdaBoostRegressor
reg = AdaBoostRegressor(random_state=0, n_estimators=100)
reg.fit(X_train,y_train)
predictions = reg.predict(X_test)
print(reg.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
adb = mean_squared_error(y_test, predictions)
adb
```

```
0.5140541719657822
```

```
19714845.10332833
```

KNeighbors Regressor

```
from sklearn.neighbors import KNeighborsRegressor
neigh = KNeighborsRegressor(n_neighbors=2)
neigh.fit(X_train,y_train)
predictions = neigh.predict(X_test)
print(neigh.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
knn = mean_squared_error(y_test, predictions)
knn
```

```
0.24045497665180704
```

```
30814777.327935223
```

Support Vector Machine

```
from sklearn.svm import SVR
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
regr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
regr.fit(X_train,y_train)
predictions = regr.predict(X_test)
print(regr.score(X_test, y_test))
from sklearn.metrics import mean_squared_error
svm = mean_squared_error(y_test, predictions)
svm
```

```
-0.09163633723024533
```

```
44287737.554449275
```

Result and Discussion

```
test = train[600:986]
test = test.select_dtypes(exclude=['object'])
#ID = test.Id
test.fillna(0,inplace=True)
#test.drop('Id',axis = 1, inplace = True)
```

```
train.head(5)
```

	Age	Diabetes	BloodPressureProblems	AnyTransplants	AnyChronicDiseases	Height	Weight	Kn
0	45	0	0	0	0	155	57	
1	60	1	0	0	0	180	73	
2	36	1	1	0	0	158	59	
3	52	1	1	0	1	183	93	
4	38	0	0	0	1	166	88	

Fig-1 : Training and Testing Dataset

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fdfa890b08>
```

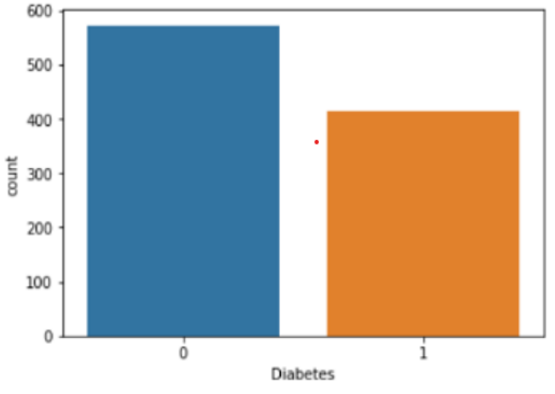


Fig-2: Dibatis problem analysis

```
sns.countplot(x="BloodPressureProblems", data = train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fdfa923808>
```

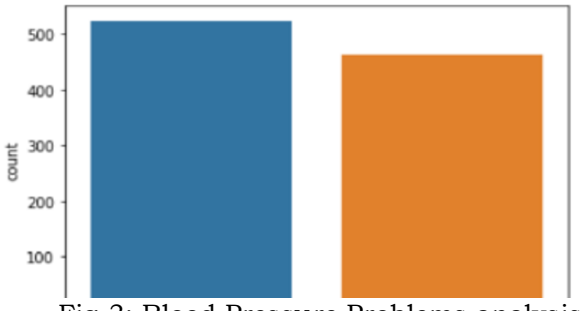


Fig-3: Blood Pressure Problems analysis

```
sns.countplot(x="KnownAllergies", data = train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fdfaa8af4
```

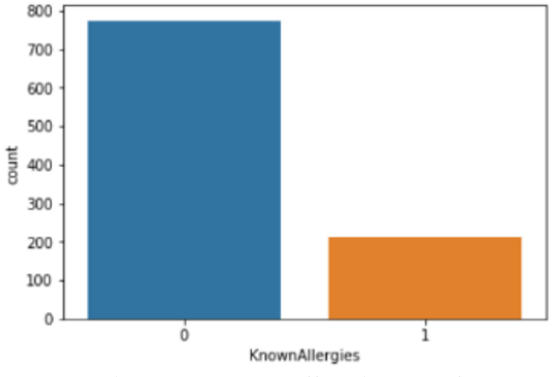


Fig- 4: Known Allergies study

```
sns.countplot(x="NumberOfMajorSurgeries", data = train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fdfab3e188>
```

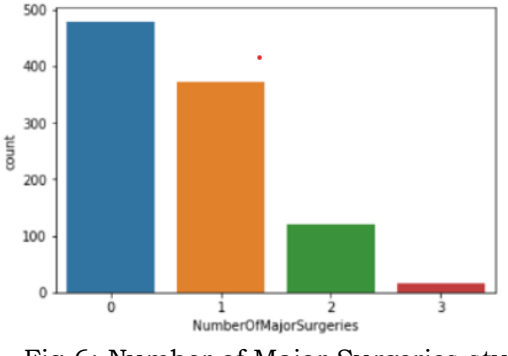


Fig-6: Number of Major Surgeries study

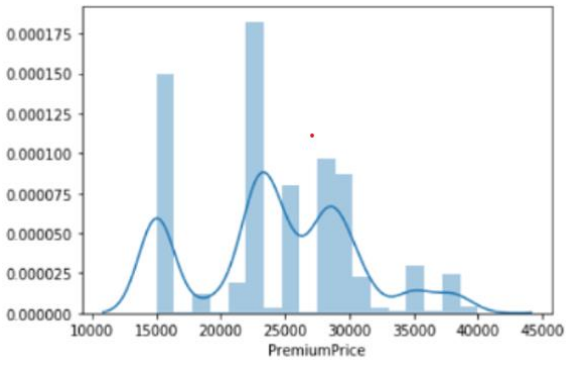


Fig-7: Correlation between the features and the label

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor

seed = 7
np.random.seed(seed)

# Model
model = Sequential()
model.add(Dense(200, input_dim=10, kernel_initializer='normal', activation='relu'))
model.add(Dense(100, kernel_initializer='normal', activation='relu'))
model.add(Dense(50, kernel_initializer='normal', activation='relu'))
model.add(Dense(25, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
```

Fig 8- Neural Network Multiview Plain code

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout

X_train, y_train = np.array(X_train), np.array(y_train)

X_train = np.reshape(X_train, [X_train.shape[0], X_train.shape[1], 1])
#print(X_train)

regressor = Sequential()
regressor.add(LSTM(units = 50, return_sequences=True, input_shape = (X_train.shape[1], 1)))

regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences=True))
regressor.add(Dropout(0.2))
```

Fig 9- Neural Network LSTM MultiView

Conclusion

To capture the heterogeneous information inside claims data, this paper provides a multi-view deep learning architecture. Separate data fields are represented as different views in our system. A feedforward neural network is used to integrate non-sequential demographic variables, an attention-based bidirectional recurrent neural network is used to capture sequential facility utilisation, and a hierarchical attention network is used to learn medical code information in the

proposed model. The attention mechanism determines the significance of input variables and interprets the projected outcome.

References

1. M. A. Morid, O. R. L. Sheng, K. Kawamoto, T. Ault, J. Dorius, and S. Abdelrahman, “Healthcare cost prediction: Leveraging fine-grain temporal patterns,” J. Biomed. Inform., vol. 91, 2019, Art. no. 103113.
2. A. S. Ash et al., “Using diagnoses to describe populations and predict costs,” Health Care Financing Rev., vol. 21, pp. 7–28, 2000.
3. M. E. Cowen, D. J. Dusseau, B. G. Toth, C. Guisinger, M. W. Zodet, and Y. Shyr, “Casemix adjustment of managed care claims data using the clinical classification for health policy research method,” Med. Care, vol. 36, pp. 1108–1113, 1998.
4. A. K. Rosen, S. A. Loveland, J. J. Anderson, C. S. Hankin, J. N. Breckenridge, and D. R. Berlowitz, “Diagnostic cost groups (DCGs) and concurrent utilization among patients with substance abuse disorders,” Health Serv. Res., vol. 37, pp. 1079–1103, 2002.
5. D. Bertsimas et al., “Algorithmic prediction of health-care costs,” Oper. Res., vol. 56, pp. 1382–1392, 2008.
6. D. O. Clark, M. Von Korff, K. Saunders, W. M. Balugh, and G. E. Simon, “A chronic disease score with empirically derived weights,” Med. Care, pp. 783–795, 1995, doi: 10.1097/00005650-199508000-00004
7. M. Von Korff, E. H. Wagner, and K. Saunders, “A chronic disease score from automated pharmacy data,” J. Clin. Epidemiol., vol. 45, pp. 197–203, 1992.
8. P. A. Fishman, M. J. Goodman, M. C. Hornbrook, R. T. Meenan, D. J. Bachman, M. C. O’Keeffe Rosetti, “Risk adjustment using automated ambulatory pharmacy data: The RxRisk model,” Med. Care, vol. 41, pp. 84–99, 2003.
9. J. P. Weiner, B. H. Starfield, D. M. Steinwachs, and L. M. Mumford, “Development and application of a population-oriented measure of ambulatory care case-mix,” Med. Care, pp. 452–472, 1991, doi: 10.1097/00005650-199105000-00006
10. Y. Zhao et al., “Measuring population health risks using inpatient diagnoses and outpatient pharmacy data,” Health Serv. Res., vol. 36, pp. 180–193, 2001.
11. Y. Zhao et al., “Predicting pharmacy costs and other medical costs using diagnoses and drug claims,” Med. Care, vol. 43, pp. 34–43, 2005.
12. C. Yang, C. Delcher, E. Shenkman, and S. Ranka, “Machine learning approaches for predicting high cost high need patient expenditures in health care,” Biomed. Eng. Online, vol. 17, no. 1, pp. 1–20, 2018, doi: 10.1186/s12938-018-0568-3
13. C.-Y. Kuo, L.-C. Yu, H.-C. Chen, and C.-L. Chan, “Comparison of models for the prediction of medical costs of spinal fusion in Taiwan diagnosis-related groups by machine learning algorithms,” Healthcare Inform. Res., vol. 24, no. 1, 2018, Art. no. 29.
14. I. Duncan, M. Loginov, and M. Ludkovski, “Testing alternative regression frameworks for predictive modeling of health care costs,” North Amer. Actuarial J., vol. 20, no. 1, pp. 65–87, 2016.

15. C. Wu, F. Wu, Y. Huang, and X. Xie, “NICE: Neural in-hospital cost estimation from medical records,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, doi: 10.1145/3357384.3358130
16. F. Wang, N. Lee, J. Hu, J. Sun, and S. Ebadollahi, “Towards heterogeneous temporal clinical event pattern discovery,” in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2012, doi: 10.1145/2339530.2339605
17. J. Zhou, F. Wang, J. Hu, and J. Ye, “From micro to macro,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2014, doi: 10.1145/2623330.2623711
18. P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh, “Deepr: A convolutional net for medical records,” IEEE J. Biomed. Health Inform., vol. 21, no. 1, pp. 22–30, Jan. 2017.
19. E. Choi, M. T. Bahadori, E. Searles, C. Coffey, and J. Sun, “Multilayer representation learning for medical concepts,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 1495– 1504.
20. C. Che, C. Xiao, J. Liang, B. Jin, J. Zho, and F. Wang, “An RNN architecture with dynamic temporal matching for personalized predictions of parkinson’s disease,” in Proc. 2017 SIAM Int. Conf. Data Mining, 2017, pp. 198–206.
21. Suryasa, I. W., Rodríguez-Gámez, M., & Koldoris, T. (2021). Get vaccinated when it is your turn and follow the local guidelines. *International Journal of Health Sciences*, *5*(3), x-xv. https://doi.org/10.53730/ijhs.v5n3.2938
22. Suryasa, I. W., Rodríguez-Gámez, M., & Koldoris, T. (2021). Health and treatment of diabetes mellitus. *International Journal of Health Sciences*, *5*(1), i-v. https://doi.org/10.53730/ijhs.v5n1.2864
23. Parmin, P., Suarayasa, K., & Wandira, B. A. (2020). Relationship between quality of service with patient loyality at general polyclinic of kamonji public health center. *International Journal of Health & Medical Sciences*, *3*(1), 86-91. https://doi.org/10.31295/ijhms.v3n1.157