INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 09 ISSUE: 01 | JAN - 2025 SJIF RATING: 8.448 ISSN: 2582-3930

Predicting Chronic Kidney Disease using MachineLearning Algorithms

[1]B. Narsimhulu, Assistant Professor, Dept of CSE,, ACE Engineering College,Hyderabad,India
bnreddy25@gmail.com

[2]Adepu Shreya,Student CSE ACE Engineering College,Hyderabad,India
shreyaadepu144@gmail.com

[3]Rachakatla pojjitha CSE ACE Engineering College,Hyderabad,India
poojitharachakatla@gmail.com

[4]Thalla Nithyananda Reddy CSE ACE Engineering College,Hyderabad,India
thallanithya@gmail.com

[5]Vorsu Karthikeya CSE ACE Engineering College,Hyderabad,India
Karthikeya2042@gmail.com

Abstract—In today's busy world, health is often neglected until symptoms appear. Chronic Kidney Disease (CKD) is particularly challenging as it often shows no symptoms, making early detection difficult and increasing the risk of severe complications. Machine learning (ML) provides a solution with its strong predictive capabilities.This study evaluated nine ML models, including KNN, Decision Tree, Random Forest, XGBoost, Stochastic Gradient Boosting, Gradient Boosting Classifier, CatBoost, Ada Boost and Extra Tree Classifier proving its effectiveness in CKD prediction.

Keywords—Kidney disease, Machine Learning Technique, Kidney disease prediction, classification algorithms

I. INTRODUCTION

Chronic Kidney Disease (CKD) is a progressive condition characterized by the gradual loss of kidney function over time. It has become a major global health issue, affecting millions of individuals worldwide. Early detection of CKD is critical to prevent severe complications, including kidney failure and cardiovascular diseases. However, the asymptomatic nature of CKD, particularly in its early stages, poses significant challenges for timely diagnosis and treatment.

In recent years, advancements in machine learning (ML) have revolutionized the healthcare industry, offering robust solutions for disease prediction and diagnosis. ML algorithms excel at analyzing complex datasets, identifying patterns, and making accurate predictions, even in cases where symptoms are minimal or absent. These capabilities make ML an ideal tool for addressing the challenges associated with CKD detection.

II. RELATED WORK

S.Gopika, et al. [8] have developed a method for predicting CKD using cluster analysis. The major goal is to use the clustering technique to identify kidney function failure. The findings of the trial showed that the Fuzzy C algorithm produces better outcomes and has an accuracy rate of 89%.

Based on an aging dataset of CKD, Deepika et al. [12] developed a project for the prediction of chronic kidney disease. 24 attributes and 1 target variable were present in the dataset. They used the KNN and Naïve Bayes supervised machine learning algorithms to develop the model. KNN and Nave Bayes both obtained accuracy levels of 91% and 97%, respectively.

Kidney function test (KFT) dataset was gathered by Vijayarani and Dhayanand [10] from medical labs, research facilities, and hospitals. The dataset included 584 occurrences, 6 attributes, and the support vector machine (SVM) and artificial neural network classifier techniques (ANN). It was discovered that ANN had the highest accuracy, reaching 87.7%.

III. PROPOSED SYSTEM

The primary goal of the proposed system is to develop a more accurate and robust predictive model for Chronic Kidney Disease (CKD) using various machine learning algorithms compared to the existing system. The aim is to facilitate early detection of CKD and improve patient outcomes.

Key Components and Approaches:

The system will utilize a range of ML algorithms including:

- Decision Tree Classifier
- Random Forest Classifier
- k-Nearest Neighbors (k-NN)
- AdaBoost
- Stochastic Gradient Boosting
- Gradient Boost Classifier
- CatBoost
- XGBoost
- Extra Trees Classifier

Comparative Analysis:

- Each of these algorithms will be trained and evaluated on the same data.
- The performance of each algorithm will be measured and compared using metrics like accuracy, precision, recall, F1-score and the area under ROC curve.
- This will help determine the best-performing model for predicting CKD.

IV. EXPLORATORY DATA ANALYSIS

Finding broad patterns in the data is the goal of exploratory data analysis, or EDA. Outliers and potentially surprising data elements are included in these patterns. In any data analysis process, EDA is a crucial initial step. Designing statistical analyses that produce insightful results can be aided by knowing the locations of outliers and the relationships between variables. Sites in biological monitoring data are probably subject to several stresses. Thus, initial explorations of stressor correlations are critical before one attempt to relate stressor variables to biological response variables. EDA can provide insights into candidate causes that should include in a causal assessment

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       391 non-null    float64
 1   blood_pressure            388 non-null    float64
 2   specific_gravity          353 non-null    float64
 3   albumin                   354 non-null    float64
 4   sugar                     351 non-null    float64
 5   red_blood_cells           248 non-null    object
 6   pus_cell                  335 non-null    object
 7   pus_cell_clumps           396 non-null    object
 8   bacteria                  396 non-null    object
 9   blood_glucose_random      356 non-null    float64
 10  blood_urea                381 non-null    float64
 11  serum_creatinine          383 non-null    float64
 12  sodium                    313 non-null    float64
 13  potassium                 312 non-null    float64
 14  haemoglobin               348 non-null    float64
 15  packed_cell_volume        330 non-null    object
 16  white_blood_cell_count    295 non-null    object
 17  red_blood_cell_count      270 non-null    object
 18  hypertension              398 non-null    object
 19  diabetes_mellitus         398 non-null    object
 20  coronary_artery_disease   398 non-null    object
 21  appetite                  399 non-null    object
 22  peda_edema                399 non-null    object
 23  aanemia                   399 non-null    object
 24  class                     400 non-null    object
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
```

```
# converting necessary columns to numerical type

df['packed_cell_volume'] = pd.to_numeric(df['packed_cell_volume'], errors='coerce')
df['white_blood_cell_count'] = pd.to_numeric(df['white_blood_cell_count'], errors='coerce')
df['red_blood_cell_count'] = pd.to_numeric(df['red_blood_cell_count'], errors='coerce')
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       391 non-null    float64
 1   blood_pressure            388 non-null    float64
 2   specific_gravity          353 non-null    float64
 3   albumin                   354 non-null    float64
 4   sugar                     351 non-null    float64
 5   red_blood_cells           248 non-null    object
 6   pus_cell                  335 non-null    object
 7   pus_cell_clumps           396 non-null    object
 8   bacteria                  396 non-null    object
 9   blood_glucose_random      356 non-null    float64
 10  blood_urea                381 non-null    float64
 11  serum_creatinine          383 non-null    float64
 12  sodium                    313 non-null    float64
 13  potassium                 312 non-null    float64
 14  haemoglobin               348 non-null    float64
 15  packed_cell_volume        329 non-null    float64
 16  white_blood_cell_count    294 non-null    float64
 17  red_blood_cell_count      269 non-null    float64
 18  hypertension              398 non-null    object
 19  diabetes_mellitus         398 non-null    object
 20  coronary_artery_disease   398 non-null    object
 21  appetite                  399 non-null    object
 22  peda_edema                399 non-null    object
 23  aanemia                   399 non-null    object
 24  class                     400 non-null    object
dtypes: float64(14), object(11)
memory usage: 78.2+ KB
```

[11]:
```
# Extracting categorical and numerical columns

cat_cols = [col for col in df.columns if df[col].dtype == 'object']
num_cols = [col for col in df.columns if df[col].dtype != 'object']
```

[13]:
```
# replace incorrect values

df['diabetes_mellitus'].replace(to_replace = {'\tno':'no','\tyes':'yes',' yes':'yes'},inplace=True)

df['coronary_artery_disease'] = df['coronary_artery_disease'].replace(to_replace = '\tno', value='no')

df['class'] = df['class'].replace(to_replace = {'ckd\t': 'ckd', 'notckd': 'not ckd'})
```

[14]:
```
df['class'] = df['class'].map({'ckd': 0, 'not ckd': 1})
df['class'] = pd.to_numeric(df['class'], errors='coerce')
```

[15]:
```
cols = ['diabetes_mellitus', 'coronary_artery_disease', 'class']

for col in cols:
    print(f"{col} has {df[col].unique()} values\n")
```

```
diabetes_mellitus has ['yes' 'no' nan] values

coronary_artery_disease has ['no' 'yes' nan] values

class has [0 1] values
```

[16]:
```
# checking numerical features distribution

plt.figure(figsize = (20, 15))
plotnumber = 1

for column in num_cols:
    if plotnumber <= 14:
        ax = plt.subplot(3, 5, plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

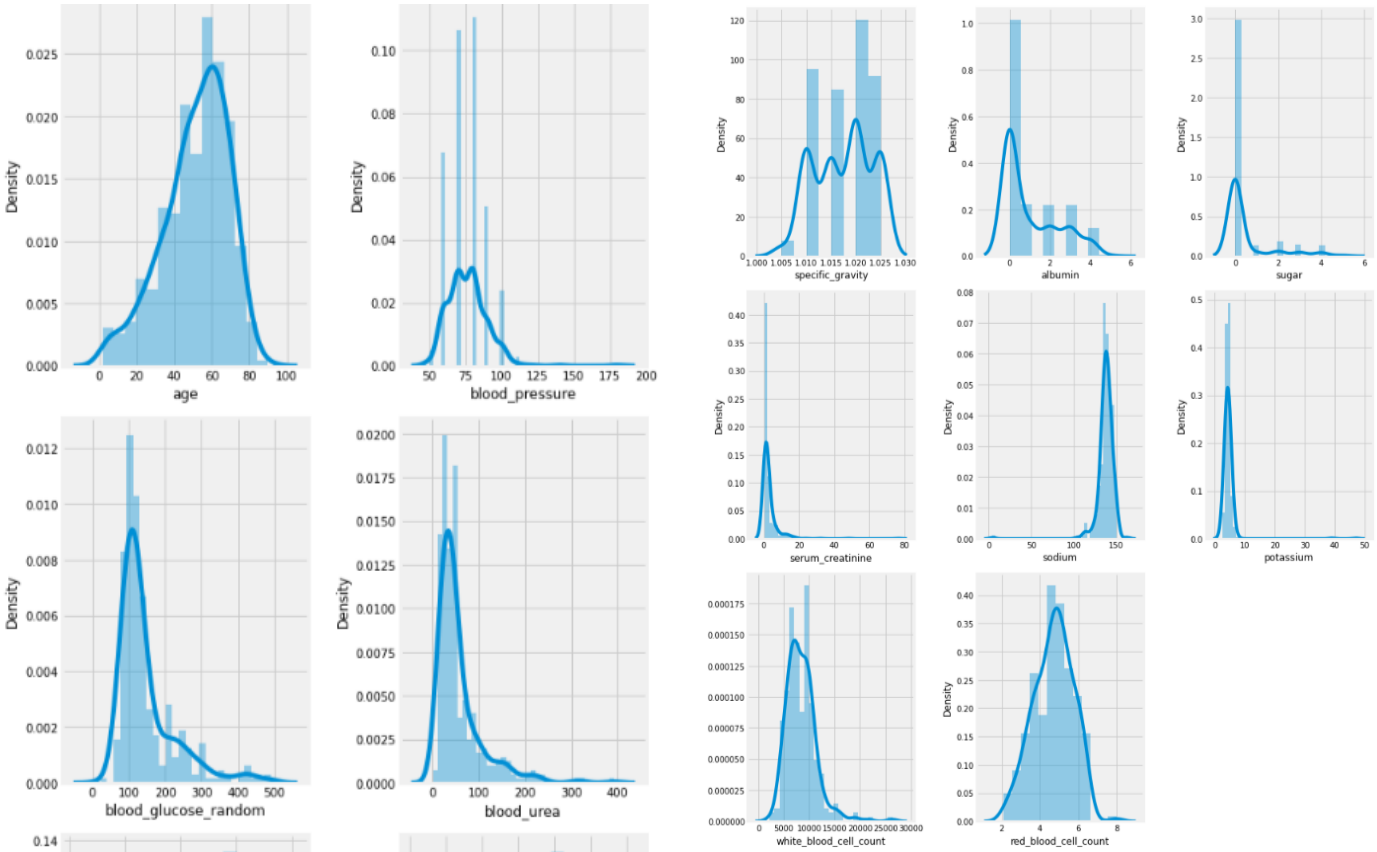


Fig: Histograms of all the indices in the data

```
# looking at categorical columns

plt.figure(figsize = (20, 15))
plotnumber = 1

for column in cat_cols:
    if plotnumber <= 11:
        ax = plt.subplot(3, 4, plotnumber)
        sns.countplot(df[column], palette = 'rocket')
        plt.xlabel(column)

    plotnumber += 1

plt.tight_layout()
plt.show()
```

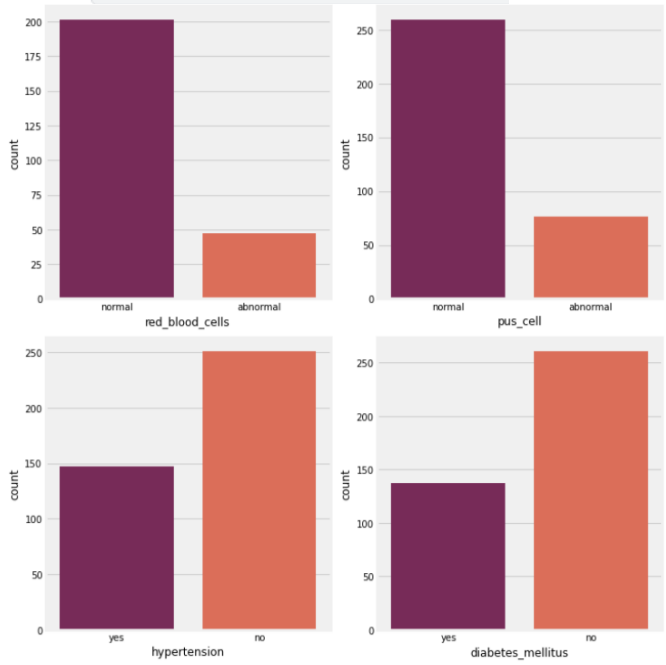
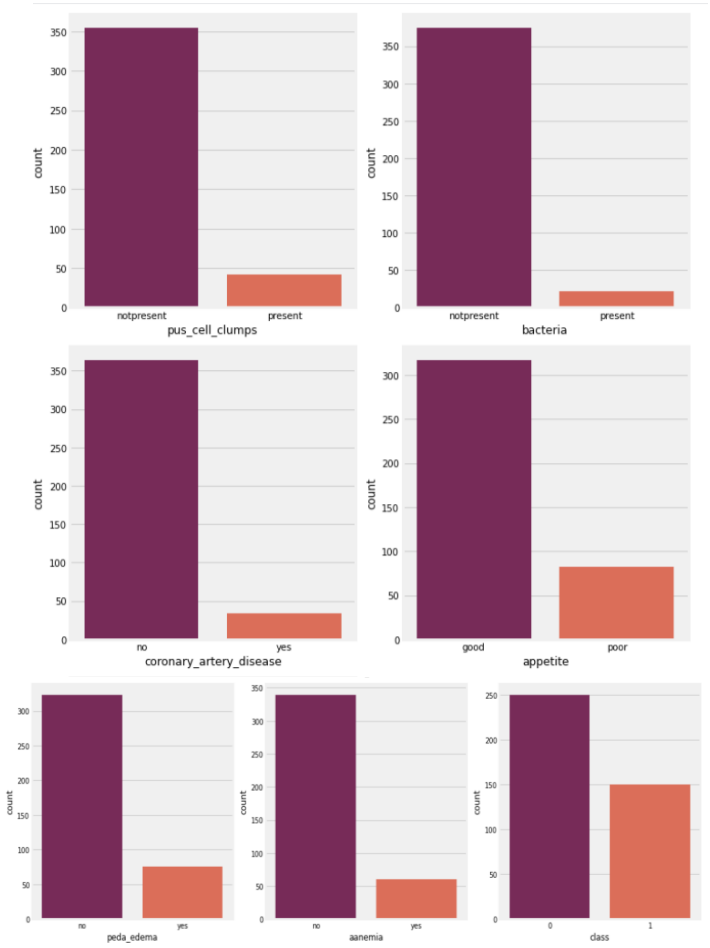


Fig: Distribution of Categorical Features in the Dataset




```
# heatmap of data

plt.figure(figsize = (15, 8))

sns.heatmap(df.corr(), annot = True, linewidths = 2, linecolor = 'lightgrey')
plt.show()
```

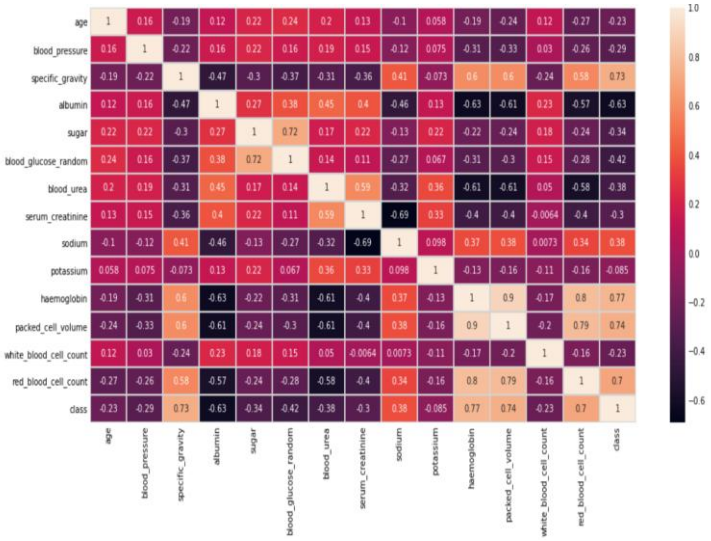


Fig: Heat map of the dataset

V. ALGORITHMS USED

- **Decision Tree Classifier**

A Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It splits the data into branches at decision nodes based on feature values, forming a tree structure. Each leaf node represents a class label or decision outcome. Decision Trees are intuitive and work well on non-linear data but are prone to overfitting.

- **Random Forest Classifier**

A Random Forest is an ensemble learning algorithm that creates multiple Decision Trees during training and combines their outputs for more accurate predictions. It reduces overfitting by averaging or voting across trees, making it robust and effective on a variety of datasets.

- **k-Nearest Neighbors (k-NN)**

The k-Nearest Neighbors (k-NN) algorithm is a simple, non-parametric method used for classification and regression. It classifies a data point based on the majority class of its k nearest neighbors (using a distance metric like Euclidean distance). It's computationally expensive for large datasets.

- **AdaBoost (Adaptive Boosting)**

AdaBoost is a boosting ensemble method that combines weak learners, typically Decision Trees, into a strong learner. It assigns higher weights to misclassified samples in subsequent iterations, improving accuracy iteratively. It's sensitive to noise and outliers.

- **Stochastic Gradient Boosting (SGB)**

Stochastic Gradient Boosting is a variation of Gradient Boosting that introduces randomness by subsampling the data before creating each tree. This reduces overfitting and improves generalization performance.

- **Gradient Boosting Classifier**

Gradient Boosting is an ensemble technique where weak learners (typically Decision Trees) are sequentially trained, with each one attempting to correct the errors of the previous ones. It minimizes a loss function by applying gradient descent, leading to a strong predictive model.

- **CatBoost**

CatBoost (Categorical Boosting) is a gradient boosting algorithm designed specifically to handle categorical features without requiring extensive preprocessing. It's fast, efficient, and avoids overfitting, making it ideal for datasets with many categorical variables.

- **XGBoost (Extreme Gradient Boosting)**

XGBoost is an optimized implementation of Gradient Boosting that is fast, efficient, and highly customizable. It incorporates techniques like regularization to reduce overfitting and handles

missing values effectively. XGBoost is widely used in competitions and real-world applications.

- **Extra Trees Classifier**

Extra Trees (Extremely Randomized Trees) is an ensemble learning algorithm similar to Random Forest but differs in the way trees are constructed. It randomly selects thresholds for splitting features, making it computationally faster and more robust to overfitting on noisy data.

VI. SOFTWARE REQUIREMENTS

- **Coding Language:** Python
- **Libraries:** Pandas, NumPy
- **Tools:** Matplotlib, Seaborn, Plotly

VII. HARDWARE REQUIREMENTS

- **Processor :** i5 or Greater
- **RAM :** 8gb
- **Storage :** 5gb free disc space

VIII.OUTPUT

The Following Results have been obtained from the evaluation of the five algorithms on the test data.

Algorithm	Accuracy
Extra Tree Classifier	97.5%
Ada Boost Classifier	96.6%
Cat Boost	96.6%
Gradient Boosting Classifier	95.8%
Stochastic Gradient Boosting	95.8%
Random Forest Classifier	95%
Decision Tree Classifier	94.1%
XG Boost	94.1%
KNN	70%

IX. CONCLUSION

The objective is to leverage machine learning techniques to accurately predict Chronic Kidney Disease (CKD). By thoroughly analyzing the dataset and applying effective preprocessing methods, critical predictors of CKD are identified. Utilizing multiple machine learning algorithms allows for a comprehensive comparison to determine the most efficient model in terms of accuracy, interpretability, and robustness.

The outcomes have the potential to integrate into clinical workflows, supporting healthcare professionals in early diagnosis, personalized treatment, and resource optimization, ultimately reducing the impact of CKD on individuals and healthcare systems.

X. REFERENCES

[1] Mahadevan, V. Anatomy of the kidney and ureter. Surgery 2019, 37, 359–364. [CrossRef]

[2] CKDPrediction Dataset. Available online: https://www.kaggle.com/datasets/abhia1999/chronic-kidney-disease (accessed on 27 June 2022).

[3] Dritsas, E., & Trigka, M. (2022). Machine learning techniques for chronic kidney disease risk prediction. Big Data and Cognitive Computing, 6(3), 98.

[4] Debal, D. A., & Sitote, T. M. (2022). Chronic kidney disease prediction using machine learning techniques. Journal of Big Data, 9(1), 1-19.

[5] Poonia, R. C., Gupta, M. K., Abunadi, I., Albraikan, A. A., Al-Wesabi, F. N., & Hamza, M. A. (2022, February). Intelligent diagnostic prediction and classification models for detection of kidney disease. In Healthcare (Vol. 10, No. 2, p. 371). MDPI. 27.

[6] Bai, Q., Su, C., Tang, W., & Li, Y. (2022). Machine learning to predict end stage kidney disease in chronic kidney disease. Scientific reports, 12(1), 1-8. 28.

[7] Ebiaredoh-Mienye, S. A., Swart, T. G., Esenogho, E., & Mienye, I. D. (2022). A machine learning method with filter-based feature selection for improved prediction of chronic kidney disease. Bioengineering, 9(8), 350.

[8] Abdel-Fattah, M. A., Othman, N. A., & Goher, N. (2022). Predicting Chronic Kidney Disease Using Hybrid Machine Learning Based on Apache Spark. Computational Intelligence and Neuroscience, 2022.

[9] Min Chen, Yixue Hao, Kai Hwang, Fellow, IEEE, Lu Wang, and Lin Wang, "Disease Prediction by Machine Learning over Big Data from Healthcare Communities", IEEE Access 2017.

[10] Garcia, Vincent & Debreuve, Eric & Barlaud, Michel. (2008). Fast k Nearest Neighbor Search using GPU. CVPR Workshop on Computer Vision on GPU. 10.1109/CVPRW.2008.4563100

[11] Jos Timanta Tarigan, C.L.G., Elviawaty Muisa Zamzami, A REVIEW ON APPLYING MACHINE LEARNING IN GAME INDUSTRY International Journal of Advanced Science and Technology, 2019-09-27 28(2).

[12] Chronic Kidney Disease Data Set, Available from: https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease

[13] D. Tian, J. Zhou, Y. Wang, Y. Lu, H. Xia, and Z. Yi, "A dynamic and self-adaptive network selection method for multimode communications in heterogeneous vehicular telematics," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3033–3049, 2015.

[14] P. Groves, B. Kayyali, D. Knott, and S. V. Kuiken, "The big data revolution in healthcare: Accelerating value and innovation," 2016.

[15] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," Mobile Networks and Applications, vol. 19, no. 2, pp. 171–209, 2014.

[16] J. C. Ho, C. H. Lee, and J. Ghosh, "Septic shock prediction for patients with missing data," ACM Transactions on Management Information Systems (TMIS), vol. 5, no. 1, p. 1, 2014.

[17] "Ictclas," http://ictclas.nlpir.org/

[18] "word2vec," https://code.google.com/p/word2vec/

[19] Y.-D. Zhang, X.-Q. Chen, T.-M. Zhan, Z.-Q. Jiao, Y. Sun, Z.-M. Chen, Y. Yao, L.-T. Fang, Y.-D. Lv, and S.-H. Wang, "Fractal dimension estimation for developing pathological brain detection system based on minkowski-bouligand method," IEEE Access, vol. 4, pp. 5937–5947, 2016.

[20] S. Basu Roy, A. Teredesai, K. Zolfaghar, R. Liu, D. Hazel, S. Newman, and A. Marinez, "Dynamic hierarchical classification for patient risk-ofreadmission," in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, 2015, pp. 1691–1700.

[21] Sak Haim, Andrew Senior, Franoise Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition", 2014.

[22] D. W. Hosmer, S. Lemeshow, Applied Logistic Regression, Wiley Interscience, 2000.

[23] C.J.C. Burges, "Simplified Support Vector Decision Rules", Proc. 13th Int'l Conf. Machine Learning, pp. 71-77, 1996.