A Minor Project Synopsis on

SP-IMLA: STROKE PREDICTION USING AN INTEGRATED MACHINE LEARNING APPROACH

Submitted to Manipal University, Jaipur

Towards the partial fulfillment for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In Computers Science and Engineering

2020-2021

By

Mritunjay Mehta | Ashima Garg

189301222 | 189301119

MANIPAL UNIVERSITY
JAIPUR

Under the guidance of

Amit Kumar Bairwa

Department of Computer Science and Engineering

School of Computing and Information Technology

Manipal University Jaipur

Jaipur, Rajasthan

March 2021

ABSTRACT

Stroke is becoming an important cause of premature death and disability in low-income and middle-income countries like India, largely driven by demographic changes and enhanced by the increasing prevalence of the key modifiable risk factors. As a result, developing countries are exposed to a double burden of both communicable and non-communicable diseases. The poor are increasingly affected by stroke, because of both the changing population exposures to risk factors and, most tragically, not being able to afford the high cost for stroke care. Majority of stroke survivors continue to live with disabilities, and the costs of on-going rehabilitation and long term-care are largely undertaken by family members, which impoverish their families. Being able to predict the possibility of a stroke is highly desirable for clinicians. This allows clinicians to set reasonable goals with patients and relatives, and to reach shared before-care decisions for recovery or rehabilitation.

The objective to do this research through literature review is to search for the research papers that similarities with the work we are aiming for and to understand the objective behind their work, the techniques they chose for implementation of their project and the reason behind choosing that techniques. This will give us a better idea to finalize our project framework as to what techniques should we use, why to use it and how to bring out the maximum efficiency in our project.

The aim of this study is to try to know more about strokes and find the best model for stroke prediction.

TABLE OF CONTENTS

1. INTRODUCTION

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths.

First what is a stroke?

- Stroke is a medical emergency. A stroke occurs when blood flow to a part of your brain is interrupted or reduced, preventing brain tissue from getting oxygen and nutrients. Brain cells begin to die within minutes.

Risk factors for having a stroke include:

- **Age:** People aged 55 years and over
- **Hypertension:** if the systolic pressure is 140 mm Hg or more, or the diastolic pressure is 90 mm Hg or more
- **Hypercholesterolemia:** If the cholesterol level in the blood is 200 milligrams per decilitre
- **Smoking**
- **Diabetes**
- **Obesity:** if the body mass index (BMI) is 30 or more

The given dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient.

1.1 MOTIVATION

Stroke is becoming an important cause of premature death and disability in low-income and middle-income countries like India, largely driven by demographic changes and enhanced by the increasing prevalence of the key modifiable risk factors. As a result, developing countries are exposed to a double burden of both communicable and non-communicable diseases. The poor are increasingly affected by stroke, because of both the changing population exposures to risk factors and, most tragically, not being able to afford the high cost for stroke care. Majority of stroke survivors continue to live with disabilities, and the costs of on-going rehabilitation and long term-care are largely undertaken by family members, which impoverish their families. Being able to predict the possibility of a stroke

is highly desirable for clinicians. This allows clinicians to set reasonable goals with patients and relatives, and to reach shared before-care decisions for recovery or rehabilitation.

The aim of this study is to try to know more about strokes and make a model for stroke prediction.

2. LITERATURE REVIEW

2.1. A STUDY OF PREVIOUS WORKS

1) **An integrated machine learning approach to stroke prediction**

- Aditya Khosla, Hsu-Kuang Chiu, Cliff Chiung-Yu Lin, Junling Hu, Yu Cao.

1. A systematic approach to impute the missing entry in the data set.
2. Based on the automatic procedure, select a relevant feature subset.
3. To evaluate the prediction performance, apply learning algorithms.

In the USA, stroke is the 3rd leading cause of death and the principal cause of serious long-term disability.

For early intervention and treatment, accurate prediction of stroke is highly valuable. In this study, for stroke prediction on the Cardiovascular Health Study (CHS) dataset, we compare the Cox proportional hazards model with a machine learning approach. Specifically, we look into the common problems of feature selection, data imputation, and prediction in the medical datasets. An automatic feature for the selection of algorithms is proposed that selects robust features based on our proposed heuristic: conservative mean. Our proposed feature selection algorithm, Combined with Support Vector Machines (SVMs), achieves a greater area under the ROC curve (AUC). This is as compared to the L1 regularized Cox feature selection algorithm and the Cox proportional hazards model. Furthermore, the concept of margin-based classifiers with censored regression to achieve a better concordance index than the Cox model is presented in a margin-based censored regression algorithm. Overall, our approach is much better than the current state-of-the-art in both concordance index and metrics of AUC. Also, our work has been reckoned as potential risk factors that haven't been discovered by traditional approaches. Where risk factors are not well understood and missing data are common,

Our method can be implemented for the clinical prediction of other diseases.

2) **A Comparative Analysis for Various Stroke Prediction Techniques**

- M. Sheetal Singh, Prakash Choudhary, Khel Chandra Thong

Mostly occurring to a person of age 65 years and above but nowadays also happen in younger age due to unhealthy diet, stroke is a major life-threatening disease. A stroke can be predicted when predicted in its early stages. In this paper, 5 different machine learning techniques were used to predict stroke on Cardiovascular Health Study (CHS) dataset. Principal Component Analysis (PCA) is used for dimension reduction, Decision Tree (DT) with the C4.5 algorithm for feature selection, and Artificial Neural Network (ANN) and Support Vector Machine (SVM) is used for classification.

Tested on different data samples based on different machine learning techniques, the predictive methods discussed in this paper are from the different methods applied, the composite method of PCA, DT, and ANN give the optimal result.

3**) Implementation of Machine Learning Model to Predict Heart Failure Disease**

- Fahd Saleh Alotaibi

Heart Failure (HF) is currently one of the most common diseases that can lead to the problematic scenario. Over a huge fraction of the total world population, around 26 million, suffer from this condition every year. It is a very complicated process to predict heart failure at right time, be it from the heart consultant or surgeon's point of view. Fortunately, with the help of various machine learning techniques and the existence of Neural Networks, which can aid the medical field and can visualize how to use the medical data efficiently. This paper aimed at improving the Heart Failure prediction accuracy using the UCI heart disease dataset. Various traditional machine learning approaches are used to understand the data and predict the Heart Failure chances in a medical database.

Upon an extended insight of the paper, the current work had increased the previous accuracy score in the prediction of heart disease. The usage of the machine learning models presented in this research with medical information systems would be very beneficial to predict Heart Failure or any other disease using the live raw data collected from patients.

2.2. **OUTCOME OF LITERATURE REVIEW**

By analyzing various research papers and comparing it with our project, we deduced the best techniques that we should use to get maximum efficiency. We studied closely the way those techniques were used in those research papers and then tried coming up with the techniques that we felt would be the best fit for our research paper.

2.3. **PROBLEM STATEMENT**

"Stroke prediction using integrating machine learning approach, applying various techniques/ algorithms, analyzing their efficiency, comparing them, and defining the best out of them."

2.4. **RESEARCH OBJECTIVE**

The objective to do this research through literature review is to search for the research papers that similarities with the work we are aiming for and to understand the objective behind their work, the techniques they chose for implementation of their project and the reason behind choosing that techniques. This will give us a better idea to finalize our project framework as to what techniques should we use, why to use it and how to bring out the maximum efficiency in our project.

3. METHODOLOGY AND FRAMEWORK

3.1. **SYSTEM ARCHITECTURE**

RAM	minimum 8-16 Gb
CPU	Intel Corei5 7th Generation or more is preferred.
STORAGE	minimum 256gb or more (1TB preferred).
OS	Linux/Windows (preferred)
INTERNET	High speed internet connectivity **required**.

3.2. **TECHNOLOGIES USED**

PYTHON | SKIT-LEARN | PANDAS | MATPLOTLIB | JUPYTER NOTEBOOK | SOME BASIC KNOWLEDGE OF DATA SCIENCE

3.3. ALGORITHMS / TECHNIQUES

3.3.1. **K-MEANS**

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into *K*pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It is a **centroid-based algorithm**, or a **distance-based algorithm**. In K-Means, each cluster is associated with a centroid.

Table 1: K-means Pros. & Cons.

Pros	Cons
Relatively simple to implement. Scales to large data sets. Guarantees convergence. Can warm-start the positions of centroids. Easily adapts to new examples. Generalizes to clusters of different shapes and sizes, such as elliptical clusters.	Choosing k manually. Being dependent on initial values. Clustering data of varying sizes and density. Clustering outliers. Scaling with number of dimensions.

3.3.2. LOGISTIC REGRESSION

Logistic regression is a statistical model that in its basic form uses a **logistic function** to model a binary dependent variable, although many more complex extensions exist. Logistic Regression is used when the dependent variable (target) is categorical.

Table 2: Logistic Regression Pros. & Cons.

Pros	Cons
Simple to implement. **Effective** **Feature scaling not needed:** Does not require input features to be scaled (can work with scaled features too, but does not require scaling) **Tuning of hyperparameters not needed.**	**Poor performance on non-linear data** (image data for e.g.) **Poor performance with irrelevant and highly correlated features** (use Boruta plot for removing similar or correlated features and irrelevant features). **Not very powerful** algorithm and can be easily outperformed by other algorithms. **High reliance on proper presentation of data**. All the important variables / features should be identified for it to work well.

3.3.3. ONE HOT ENCODING

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a **better job in prediction**.

4. WORK DONE

4.1. DETAILS / PLANNING

1. First, we began with **Cleaning of The Data** to make it efficient.
2. Next by **Plotting and Analysing the Data** we formed our observations on basis of gender, hypertension, heart disease, married or not, residence type, job type, age, glucose level and BMI.
3. Through **One-Hot Encoding** we converted the string values into Integer values to have the complete data in integer form.
4. To have effective results, we **balanced the data** to have equal samples of people who had and did not have strokes and drop all the other.
5. On the resultant dataset, we applied **Logistic Regression** and compared as to how effective our results are.
6. For better comparison and analysis in the future, we created the **Confusion Matrix** for logistic regression.
7. On the dataset, we further applied **K-Means Clustering** and compared as to how effective our results are.
8. For better comparison and analysis in the future, we created the **Clustering Report** for K-Means Clustering.
9. We further created **Confusion Matrix** for better comparison and analysis.

4.2. RESULTS AND DISCUSSION

1. **Cleaning of The Data:** We removed those specific parts of the data where BMI was mentioned as zero or not available and removing the indexing column.

2. **Plotting and Analysing the Data:** Through this we figured out what all factors contribute to the final decision as to if a person experiences a stroke or not and how much each category of these factors affects.

3. **One-Hot Encoding:** We converted categorical variables (i.e., **gender, smoking_status, work_type, Residence_type, ever_married**) to binary encoded variables.

```
In [17]: gender = pd.get_dummies(df['gender'], prefix='gender')
         smoking_status = pd.get_dummies(df['smoking_status'], prefix='smoking_status')
         work_type = pd.get_dummies(df['work_type'], prefix='work_type')
         Residence_type = pd.get_dummies(df['Residence_type'], prefix='Residence_type')
         ever_married = pd.get_dummies(df['ever_married'], prefix='ever_married')
         df_final = pd.merge(df_final, gender, left_index=True, right_index=True)
         df_final = pd.merge(df_final, work_type, left_index=True, right_index=True)
         df_final = pd.merge(df_final, smoking_status, left_index=True, right_index=True)
         df_final = pd.merge(df_final, Residence_type, left_index=True, right_index=True)
         df_final = pd.merge(df_final, ever_married, left_index=True, right_index=True)
         df final
```

Figure 1: One-hot encoding **I**

Out[17]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	gender_Female	gender_Male	gender_Other	work_type_Govt_job	...	work_type_ empl
0	67.0	0	1	228.69	36.6	1	0	1	0	0	...	
2	80.0	0	1	105.92	32.5	1	0	1	0	0	...	
3	49.0	0	0	171.23	34.4	1	1	0	0	0	...	
4	79.0	1	0	174.12	24.0	1	1	0	0	0	...	
5	81.0	0	0	186.21	29.0	1	0	1	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
5104	13.0	0	0	103.08	18.6	0	1	0	0	0	...	
5106	81.0	0	0	125.20	40.0	0	1	0	0	0	...	
5107	35.0	0	0	82.99	30.6	0	1	0	0	0	...	
5108	51.0	0	0	166.29	25.6	0	0	1	0	0	...	
5109	44.0	0	0	85.28	26.2	0	1	0	0	1	...	

4909 rows × 22 columns

Figure 2: One-hot encoding **II**

4. **Balancing the data:** To have effective results, we **balanced the data** to have equal samples of people who had and did not have strokes and drop all the other.

```
stroke1=list(df['stroke'])
count=0
for i in range(len(stroke1)):
    if count>211:
        if stroke1[i]==0:
            stroke1[i]=2
    if stroke1[i]==0:
        count+=1
df_final=df_final.drop(['stroke'],axis=1)
df_final['stroke']=stroke1
df_final= df_final[df_final['stroke'] != 2]
y_train=np.array(list(df_final['stroke']))
df_final=df_final.drop(['stroke'],axis=1)
```

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

Figure 3: Balancing the data **I**

In [20]:
```
df_final
```

Out[20]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	gender_Female	gender_Male	gender_Other	work_type_Govt_job	work_t
0	67.0	0	1	228.69	36.6	0	1	0	0	
2	80.0	0	1	105.92	32.5	0	1	0	0	
3	49.0	0	0	171.23	34.4	1	0	0	0	
4	79.0	1	0	174.12	24.0	1	0	0	0	
5	81.0	0	0	186.21	29.0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	
459	11.0	0	0	87.51	24.4	1	0	0	0	
460	7.0	0	0	72.35	17.0	1	0	0	0	
461	16.0	0	0	113.47	19.5	1	0	0	0	
462	44.0	0	0	103.78	49.8	1	0	0	0	
463	78.0	0	0	115.43	27.8	1	0	0	0	

421 rows × 21 columns

Figure 4: Balancing the data **I**

5. Logistic Regression

```
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
count=0
for i in range(len(y_test)):
    if y_pred[i]==y_test[i]:
        count+=1
print(count*100/len(y_pred))
```

```
94.60285132382892
```

Figure 4: Logistic Regression

Accuracy came out to be **94.60%**.

6. Confusion Matrix for Logistic Regression:

```
from sklearn.metrics import confusion_matrix
res=confusion_matrix(y_train, y_test)
res
```

```
array([[153,  59],
       [ 46, 163]])
```

Figure 5: Confusion Matrix **I**

```
Out[55]: array([[153,  59],
                [ 46, 163]])

In [56]: TN=res[1][1]
         TP=res[0][0]
         FN=res[1][0]
         FP=res[0][1]
         print(TP,FP,FN,TN)

         153 59 46 163

In [57]: Sensitivity=(TP)/(TP+FN)
         Specificity=(TN)/(TN+FP)
         Precision=(TP)/(TP+FP)
         Recall=(TP)/(TP+FN)
         f1=2*(Precision*Recall)/(Precision+Recall)

In [58]: print('Sensitivity: '+str(Sensitivity))
         print('Specificity: '+str(Specificity))
         print('Precision: '+str(Precision))
         print('Recall: '+str(Recall))
         print('f1: '+str(f1))

         Sensitivity: 0.7688442211055276
         Specificity: 0.7342342342342343
         Precision: 0.7216981132075472
         Recall: 0.7688442211055276
         f1: 0.7445255474452555
```

Figure 6: Confusion Matrix **II**

Sensitivity came out to be **0.7688**

Specificity came out to be **0.7342**

Precision came out to be **0.7217**

Recall came out to be **0.7688**

F1 came out to be **0.7445**

7. **K-means**

```
# K-means Clustering

import sklearn
from sklearn.cluster import KMeans

sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df_final[columns],y_train)
    sse.append(km.inertia_)
```

Figure 7: K-Means: finding k

```
#elbow plot
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

```
[<matplotlib.lines.Line2D at 0x1a86a9a1ac0>]
```

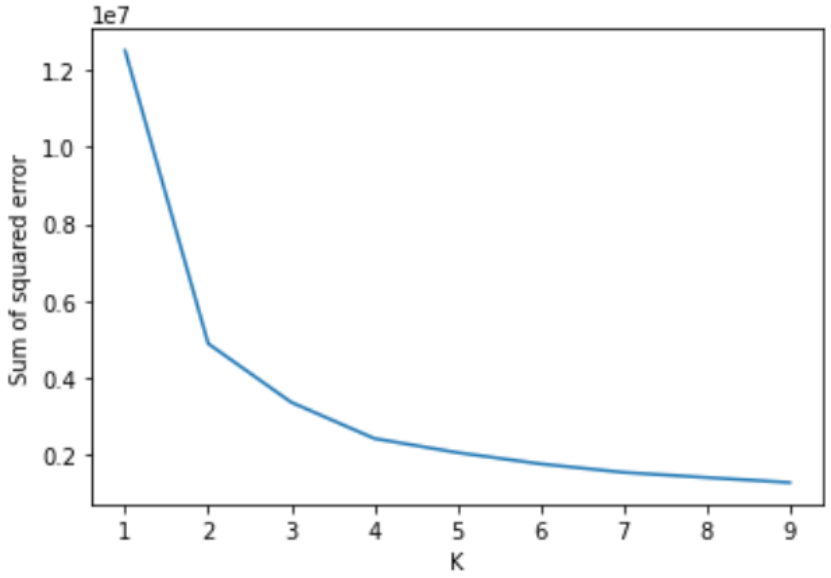

Figure 8: K-Means: Elbow Plot

```
KMeans_Clustering = KMeans(n_clusters =2, random_state=42)
KMeans_Clustering.fit(X_train)
```

```
KMeans(n_clusters=2, random_state=42)
```

```
print(KMeans_Clustering.cluster_centers_)
```

```
[[ 4.02612589e+01  6.73990499e-02  3.35510689e-02  8.94827821e+01
   2.82233670e+01  6.01247031e-01  3.98456057e-01  2.96912114e-04
   1.27375297e-01  5.34441805e-03  5.72743468e-01  1.41627078e-01
   1.52909739e-01  3.22149644e-01  1.55878860e-01  3.72624703e-01
   1.49346793e-01  4.94952494e-01  5.05047506e-01  3.80938242e-01
   6.19061758e-01]
 [ 5.78120215e+01  2.45080501e-01  1.34168157e-01  2.00527406e+02
   3.24576029e+01  5.40250447e-01  4.59749553e-01 -3.79470760e-19
   1.52057245e-01  1.78890877e-03  5.84973166e-01  2.30769231e-01
   3.04114490e-02  1.80679785e-01  2.45080501e-01  4.27549195e-01
   1.46690519e-01  4.97316637e-01  5.02683363e-01  1.37745975e-01
   8.62254025e-01]]
```

Figure 9: K-Means: Model Training & Cluster Centres

8. **Classification Report for K-Means Clustering:**

```
#prediction using kmeans and accuracy
kpred = KMeans_Clustering.predict(X_test)
print('Classification report:\n\n', sklearn.metrics.classification_report(y_test,kpred))
```

```
Classification report:

              precision    recall  f1-score   support

           0       0.96      0.87      0.91       929
           1       0.12      0.30      0.17        53

    accuracy                           0.84       982
   macro avg       0.54      0.58      0.54       982
weighted avg       0.91      0.84      0.87       982
```

Figure 10: Classification Report

9. **Confusion Matrix for K-Means Clustering:**

```
res1=confusion_matrix(y_test, kpred)
res1
```

```
array([[806, 123],
       [ 37,  16]], dtype=int64)
```

```
TN=res1[1][1]
TP=res1[0][0]
FN=res1[1][0]
FP=res1[0][1]
print(TP,FP,FN,TN)
```

```
806 123 37 16
```

Figure 11: Confusion Matrix **I**

```
Sensitivity=(TP)/(TP+FN)
Specificity=(TN)/(TN+FP)
Precision=(TP)/(TP+FP)
Recall=(TP)/(TP+FN)
Accuracy=(TP+TN)/(TP+TN+FP+FN)
f1=2*(Precision*Recall)/(Precision+Recall)

print('Sensitivity: '+str(Sensitivity))
print('Specificity: '+str(Specificity))
print('Precision: '+str(Precision))
print('Recall: '+str(Recall))
print('f1: '+str(f1))
print('Accuracy: '+str(Accuracy))
```

```
Sensitivity: 0.9561091340450771
Specificity: 0.11510791366906475
Precision: 0.8675995694294941
Recall: 0.9561091340450771
f1: 0.909706546275395
Accuracy: 0.8370672097759674
```

Figure 12: Confusion Matrix **II**

Sensitivity came out to be **0.9561**

Precision came out to be **0.8675**

Recall came out to be **0.9561**

F1 came out to be **0.9097**

Accuracy came out to be **83.71%**.

4.3. INDIVIDUAL CONTRIBUTION

ASHIMA GARG

Exploratory Data Analysis.

One-hot encoding.

MRITUNJAY MEHTA

Logistic Regression

Confusion Matrix

5. CONCLUSION AND FUTURE PLAN

Further, we would be applying more techniques and compare the results to find the best fit. To name one, we would be using K-means algorithm approach to find out the result.

6. BIBLIOGRAPHY / REFERENCES

[1] https://www.kaggle.com/fedesoriano/stroke-prediction-dataset
[2] https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages
[3] https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6
[4] https://dl.acm.org/doi/abs/10.1145/1835804.1835830
[5] https://link.springer.com/chapter/10.1007/978-981-15-4018-9_9
[6] https://thesai.org/Downloads/Volume10No6/Paper_37-
[7] Implementation_of_Machine_Learning_Model.pdf