Journal of Applied Information Science
7 (1), June 2019, 11-15
http://www.publishingindia.com/jais

Advanced Medical Diagnosis and Prediction Using Deep Learning

Hasna Hashim[1], Nayama Grace Mathew[1], K. Sabira[2], A. Nizamudeen[3] and Jithin Jacob[4*]
[1]Student, Department of CSE, UKF College of Engineering & Technology, Kerala, India.
[2]Student, Department of CSE, UKF College of Engineering & Technology, Kerala, India. Email: sabirak1997@gmail.com
[3]Student, Department of CSE, UKF College of Engineering & Technology, Kerala, India. Email: nizamial09@gmail.com
[4]Assistant Professor, Department of CSE, UKF College of Engineering & Technology, Kerala, India.
Email: jithu7771@gmail.com
*Corresponding Author

Abstract: A significant part of human life is medical diagnosis. The correct prediction of a disease revolves around various steps. Due to the tremendous advancement in the medical field, the data is quite large, which makes it challenging for medical diagnosis and prediction. Mobile health applications are becoming increasingly used as medical diagnosis. The use of Electronic Health Record (EHR) by hospitals is also increasing. These systems have many advantages such as easier access to patient data, structured information and support for decision making and better access to information. This paper focus on such a mobile application that enables the users to predict or diagnose medical conditions based on the symptoms provided by the user.

Keywords: Android, Decision tree, Flask, Machine learning, Medical diagnosis.

I. Introduction

The healthcare sector differs completely from other industries. It is a high priority sector and, irrespective of cost, people expect the highest level of care and services. It also provides exciting solutions for medical imaging following the success of deep learning in other real world applications and is seen as a key method for future health sector applications. Deep learning will not only help to not only help to select and extract features but also construct new ones, furthermore, it does not only diagnose the disease but also measure predictive target and provides actionable prediction models to help physician efficiently.

Machine learning algorithms were from the very beginning designed and used to analyze medical datasets. Today, machine learning provides several indispensable tools for intelligent data analysis. Especially in the last few years, the digital revolution provided relatively inexpensive and available means to collect and store the data.

Modern hospitals are well equipped with monitoring and other data collection devices, and data is gathered and shared in large information system. Machine learning technology is currently well suited for analyzing medical data, and in particular there is a lot of work done in medical diagnosis in small specialized diagnostic problems.

Data about correct diagnosis are often available in the form of medical records in specialized hospitals or departments. All that has to be done is to input the patient records with known correct diagnosis into a computer program to run a learning algorithm. This is of course an over simplification of the case, but this can be done by the diagnosis cases of the past.

II. Objectives

The primary objective of the system is to make the diagnosis of a patient easier by providing the previous health records accessible to all the doctors who are observing that patient. The system acts as an intermediary between the doctor and user. The objectives include improving the accuracy of existing technologies and the ability to integrate with existing systems while providing a fast and efficient user experience for all concerned parties.

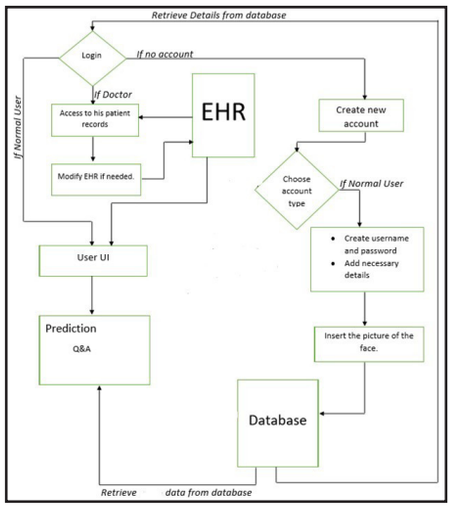

Fig. 1: Data Flow Diagram

Diagnostic apps have enormous potential to provide access to diagnostic definitions and could appeal to health care professionals as they can invite patients for early diagnosis and to patients themselves through a diagnostic adjunct [1].

III. Approaches and Methodology

A. Android Based UI Design

For mobile applications, Android defines an additional component-based framework, which includes various numbers and components in each application. The components of the activity form the basis on which the user interface operates [2]. Android is an integrated open platform provided by Google Inc. for mobile devices. It encompasses of operating system, middleware, along with some of the key applications. It has an amazing environment for development and debugging. Android provides a straightforward XML vocabulary that matches the view classes and subclasses, such as those for widgets and layouts. An android application can also programmatically create objects for view and view group.

The system's user interface design is implemented on the basis of an application's requirement. It covers the basic elements that make up a screen, how to define a screen in XML and load it into the code and

numerous other tasks that users need to handle for the user interface [3]. The application requires a login page where both user and doctor can login using the given credentials email and password, a signup page for registration of new user account, a profile view of the user, which consist of two tabs one showing the EHR of the user and the other an interface for the user to input the symptoms, a profile view of the doctor which consist of a list of all the patients. A search option is provided in the menu of the user profile view for searching nearby doctors to connect with them.

B. ML Model Design

i) Data Cleaning

Data cleaning is part of data pre-processing before data mining, prior to process of mining information in a data warehouse, data cleaning is crucial because of garbage in and garbage out principle. Data cleaning is also called data cleansing or scrubbing deals with detecting and removing errors and inconsistencies from data in order to improve quality of data. The main objective of data cleaning is to reduce the time and complexity of mining process and increase the quality of datum in data warehouse. Data quality problems can be single source problems or multisource problems [4].

Pandas is a popular python package used for data cleaning, and with good reason: It offers powerful, expressive and flexible data structures and functions that make data manipulation and analysis easy, among many other things. Data mining entails the use of computer applications for applying various learning algorithms that identify patterns within the dataset [5]. The Data frame is one of these structures. Data frame is an object provided by Pandas for data manipulation with integrated indexing. It is two-dimensional size- mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). The untidy dataset is shown in Fig. 2. Algorithm 1 and 2 is used to convert it into the form shown in Fig. 3.

Disease	Count of D	Symptom
UMLS:C00	3363	UMLS:C0008031_pain chest
		UMLS:C0392680_shortness of breath
		UMLS:C0012833_dizziness
		UMLS:C0004093_asthenia
		UMLS:C0085639_fall
		UMLS:C0039070_syncope
		UMLS:C0042571_vertigo
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0030252_palpitation
		UMLS:C0027497_nausea
		UMLS:C0002962_angina pectoris
		UMLS:C0438716_pressure chest
UMLS:C00	1421	UMLS:C0032617_polyuria
		UMLS:C0085602_polydypsia
		UMLS:C0392680_shortness of breath
		UMLS:C0008031_pain chest
		UMLS:C0004093_asthenia
		UMLS:C0027497_nausea
		UMLS:C0085619_orthopnea
		UMLS:C0034642_rale
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0241526_unresponsiveness

Fig. 2: Un-cleaned Data

Algorithm 1

```
disease_list = []

def return_list(disease):
    disease_list = []
    match = disease.replace('^','_').split('_') ctr =
    1
    for group in match: if
        ctr%2==0:
            disease_list.append(group) ctr =
        ctr + 1
    return disease_list

with open("Scraped-Data/dataset_uncleaned.csv") as csvfile:
    reader = csv.reader(csvfile)
    disease="" weight = 0
    disease_list = []
    dict_wt = {}
    dict_=defaultdict(list) for
    row in reader:
        if row[0]!="\xc2\xa0" and row[0]!="":
            disease = row[0]
            disease_list = return_list(disease)
            weight = row[1]

        if row[2]!="\xc2\xa0" and row[2]!="": symptom_list
            = return_list(row[2])

            for d in disease_list:
                for s in symptom_list:
                    dict_[d].append(s)
                dict_wt[d] = weight

    #print (dict_)
# Writing our cleaned data
 for key,values in dict_.items(): for v
        in values:
            #key = str.encode(key)
            key = str.encode(key).decode('utf-8') #.strip()
            #v = v.encode('utf-8').strip() #v =
            str.encode(v)
            writer.writerow([key,v,dict_wt[key]])

columns = ['Source','Target','Weight'] data
= pd.read_csv("Scraped-
Data/dataset_clean.csv",names=columns, encoding ="ISO-
8859-1")
data.to_csv("Scraped-Data/dataset_clean.csv",index=False)
```

Algorithm 2

```
slist=[]
dlist = []
with open("Scraped-Data/nodetable.csv","w") as
csvfile:
    writer = csv.writer(csvfile)

    for key,values in dict_.items():
        for v in values:
            if v not in slist:
                writer.writerow([v,v,"symptom"])
                slist.append(v)
        if key not in dlist:
            writer.writerow([key,key,"disease"])
            dlist.append(key)
nt_columns = ['Id','Label','Attribute']
nt_data = pd.read_csv("Scraped-
Data/nodetable.csv",names=nt_columns, encoding
="ISO-8859-1",)
nt_data.head()
nt_data.to_csv("Scraped-
Data/nodetable.csv",index=False
```

Disease	Count of D	Symptom
UMLS:C00	3363	UMLS:C0008031_pain chest
		UMLS:C0392680_shortness of breath
		UMLS:C0012833_dizziness
		UMLS:C0004093_asthenia
		UMLS:C0085639_fall
		UMLS:C0039070_syncope
		UMLS:C0042571_vertigo
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0030252_palpitation
		UMLS:C0027497_nausea
		UMLS:C0002962_angina pectoris
		UMLS:C0438716_pressure chest
UMLS:C00	1421	UMLS:C0032617_polyuria
		UMLS:C0085602_polydypsia
		UMLS:C0392680_shortness of breath
		UMLS:C0008031_pain chest
		UMLS:C0004093_asthenia
		UMLS:C0027497_nausea
		UMLS:C0085619_orthopnea
		UMLS:C0034642_rale
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0241526_unresponsiveness

Fig. 3: Cleaned Data

ii) Label Encoding

The decision tree algorithm for training the data is performed on numerical values and thus it is essential for converting the dataset into a suitable format. Label encoder encodes labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier. The categorical value is converted into numeric values. The result is shown in fig. 4.

Algorithm 3

```
data = pd.read_csv("Scraped-Data/dataset_clean.csv",
encoding ="ISO-8859-1")
data.head()

len(data['Source'].unique())
len(data['Target'].unique())

df = pd.DataFrame(data)

df_1 = pd.get_dummies(df.Target)

df_s = df['Source']

df_pivoted = pd.concat([df_s,df_1], axis=1)

df_pivoted.drop_duplicates(keep='first',inplace=True)

df_pivoted[:5]

len(df_pivoted)
cols = df_pivoted.columns
cols = cols[1:]
df_pivoted = df_pivoted.groupby('Source').sum()
df_pivoted = df_pivoted.reset_index()
df_pivoted[:5]

len(df_pivoted)
df_pivoted.to_csv("Scraped-Data/df_pivoted.csv")
```

Source	Heberden	Murphy's	Stahli's lir	abdomen	abdomina	abdomina	abnormal	abnormall	abortion	a
Alzheimer's disease	0	0	0	0	0	0	0	0	0	
HIV	0	0	0	0	0	0	0	0	0	
PneumocystisÃ‚Â carinii	0	0	0	0	0	0	0	0	0	
accidentÃ‚Â cerebrovasc	0	0	0	0	0	0	0	0	0	
acquiredÃ‚Â immuno-de	0	0	0	0	0	0	0	0	0	
adenocarcinoma	0	0	0	0	0	0	0	0	0	
adhesion	0	0	0	0	0	0	0	0	0	
affect labile	0	0	0	0	0	0	0	0	0	
anemia	0	0	0	0	0	0	0	0	0	
anxiety state	0	0	0	0	0	0	0	0	0	
aphasia	0	0	0	0	0	0	0	0	0	
arthritis	0	0	0	0	0	0	0	0	0	
asthma	0	0	0	0	0	0	0	0	0	
bacteremia	0	0	0	0	0	1	0	0	0	
benign prostatic hypertrc	0	0	0	0	0	0	0	0	0	
biliary calculus	0	0	0	0	0	0	0	0	0	
bipolar disorder	0	0	0	0	0	0	0	0	0	
bronchitis	0	0	0	0	0	0	0	0	0	
candidiasis	0	0	0	0	1	1	0	0	0	
carcinoma	0	0	0	0	0	0	0	0	0	
carcinoma breast	0	0	0	0	0	0	0	0	0	
carcinoma colon	0	0	0	0	0	0	0	0	0	

Fig. 4: Encoded Dataset

C. Model Training Using Decision Tree

As a result of the large amount of data accessible in the medical field, the decision-making process is becoming more complicated for groups and individuals developing them. This is where the need occurs for a smart decision support methodology. The decision support system should be able to

process these huge data and help health professionals to make their decisions stress-free and more dependable in order to avoid patient mismanagement and erroneous diagnosis [6].

Decision tree is among one of the most common, coherent, efficient, powerful and popular classification and prediction techniques used in machine learning and decision-making analysis. It is a straightforward, quantifiable and effective method for decision-making under circumstances of uncertainty with a simple illustration of the information gathered.

A decision tree is a method that can help in making good choices, especially decisions that involve high costs and risks. Decision trees use a graphical approach to compare and assign values to rival alternatives, combining ambiguities, costs and payoffs into a number of specific values that makes the decision making process faster and accurate.

Algorithm 4

```
df_pivoted=pd.read_csv("Scraped-Data/df_pivoted.csv")
df_pivoted.drop_duplicates(keep='first',inplace=True)


cols = df_pivoted.columns cols

= cols[2:]

df_pivoted = df_pivoted.groupby('Source').sum()
df_pivoted = df_pivoted.reset_index()

x = df_pivoted[cols]
y = df_pivoted['Source']

x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.33,
random_state=42)
mnb = MultinomialNB()
mnb = mnb.fit(x_train, y_train)

mnb.score(x_test, y_test)

# ### Inferences on train and test split

mnb_tot = MultinomialNB()
mnb_tot = mnb_tot.fit(x, y)
mnb_tot.score(x, y)
disease_pred = mnb_tot.predict(x)
disease_real = y.values
for i in range(0, len(disease_real)):
    if disease_pred[i]!=disease_real[i]:
        print ('Pred: {0} Actual:{1}'.format(disease_pred[i],
disease_real[i]))
import pickle

# These are the predicted versus actual diseases that our classifier
misclassifies.

# ### Training a decision tree

print ("DecisionTree")
dt = DecisionTreeClassifier()
clf_dt=dt.fit(x,y)
pickle.dump(clf_dt, open('model.pkl','wb'))
model = pick-+le.load( open('model.pkl','rb'))
print ("Acurracy: ", clf_dt.score(x,y))
model.predict(x_test)
```

A tedious task is the selection of suitable activation function for training the data. There is a natural choice of both output unit activation function and matching error function, according to the type of problem being solved. For regression we use linear outputs and a sum-of-squares error, for (multiple independent) binary classifications we use logistic sigmoid outputs and a cross-entropy error function, and for multiclass classification we use softmax outputs with the corresponding multiclass cross-entropy error function [4]. For classification problems involving two classes, we can use a single logistic sigmoid output, or alternatively we can use a network with two outputs having a softmax output activation function [7].

The algorithm starts by reading the dataset into a data frame using Pandas library. Then it splits the dataset into training and validation sets. The model is trained based on the training dataset and its efficiency can be calculated or cross verified using the validation set. The model can't work on unseen data because it has never seen such a disease before. Also, there is only one class label for each disease and hence no point for this. So we need to train the model entirely. Then what will we test it on? Missing data? Say given one symptom what is the disease? This is again multi-label classification. We can work symptom on symptom. What exactly is differential diagnosis, we need to replicate that [8].

D. Server Design

Flask solves the issue of connecting the application to the ML model. Flask is a python-written micro web framework. It has no abstraction layer in a database, verified form or any other components in which pre- existence third-party libraries provide common features. There is no abstraction layer in the database, no form verification or any other modules. Flask supports extensions which can add software functions, however, as if they were implemented in flask. For object-relational mappers there are extensions, form validation, upload handling, several open verification technologies, and a number of tools relating to the general framework. Extensions are much more regularly updated than the flask core program.

Flask acts as an interface that connects the application with the decision tree model. It is used to create a server which takes the values and details given by the user in JSON format and feeds these data into the model for processing it, the trained model then makes the prediction based on this data and return the result as JSON file to the flask server, the server sends back the result to the android application and it prints the data on the screen using the xml view groups and java code for the user to view it. The framework consist of a response from the android device sending the data for feeding the model, are request send from the server to the model and a response from the model to the server.

```
app = Flask(__name__)
output=[]
# Load the model
model = pickle.load(open('model.pkl','rb'))


@app.route('/api',methods=['POST'])
def predict():
    da=pd.read_csv('C:/Users/theLEGEND/Desktop/Predicting-Diseases-From-Symptoms-r
    # Get the data from the POST request.
    data = request.get_json(force=True)

    # Make prediction using model loaded from disk as per the data.
    prediction = model.predict([np.array(data['sym'])])
    list=[]
    list=data['sym']
    da.append([np.array(data['sym'])],ignore_index=True)
    add=da['Source']
    listadd=add.values.tolist()
    c=[]

    for i in range(len(da.index)+1):
        c.append(i)


    # Take the first value of prediction
    output=prediction[0]
    listadd.append(output)
    dropped=da.drop(['Unnamed: 0','Source'], axis=1)
    tobel=dropped.append(pd.Series(list,index=dropped.columns),ignore_index=True)
    tobel.insert(0,"",c)
    tobel.insert(1,"Source",listadd,True)


    export_csv=tobel.to_csv (r'C:/Users/theLEGEND/Desktop/Predicting-Diseases-Fro
    return jsonify(output)

if __name__ == '__main__':

    app.run(host='0.0.0.0',port=5002, debug=True)
```

Fig. 5: Creating a Server Using Flask

IV. Conclusion

The historical development of machine learning and its applications in medical diagnosis shows that from simple and straightforward to use algorithms, systems and methodology have emerged that enable advanced and sophisticated data analysis. In the future, intelligent data analysis will play even a more important role due to the huge amount of information produced and stored by modern technology. Current machine learning algorithms provide tools that can significantly help medical practitioners to reveal interesting relationship in their data [8].

We have experimented with the decision tree algorithm on a comparatively small dataset to produce much better result. The accuracy of the model is improved by updating the dataset by each analysis made by each user in the application. We have provided a platform for the doctor is to correct the results provided by the model, so that it could also be fed into the dataset to improve the accuracy.

References

[1] A. Jutel, and D. Lupton, "Digitizing diagnosis: A review of mobile applications in the diagnostic process," *Diagnosis*, vol. 2, no. 2, pp. 89-96, 2015.

[2] T. Bläsing, L. Batyuk, A. D. Schmidt, S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox system for suspicious software detection," *IEEE 5th Int. Conf. on Malicious and Unwanted Software*, France, 2010.

[3] R. A. Soni, "A study paper on Android UI," *International Journal of Enterprise Computing and Business Systems*, vol. 2, no. 1, 2013.

[4] S. Devi, and A. Kalia, "Study of data cleaning & comparison of data cleaning tools," *International Journal of Computer Science and Mobile Computing*, vol. 4, no. 3, pp. 360-370, 2015.

[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493-2537, 2011.

[6] J. N. Mamman, M. B. Abdullahi, J. KoloAlhassan, and S. A. Adepoju, "On the use of decision tree for treatment options in medical decision," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 2, pp. 71-78, 2015.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[8] I. Kononenko, "Machine learning for medical diagnosis: History, state of the art perspective," *Artificial Intelligence in Medicine*, vol. 23, no. 1, pp. 89-109, 2001.