SMART HEALTH PREDICTION SYSTEM

Submitted by
Madhav Somanath
mad00154

ABSTRACT

It might have happened so many times that you or someone yours need doctors help immediately, but they are not available due to some reason. The Health Prediction system is an end user support and online consultation project. Here we propose a system that allows users to get instant guidance on their health issues through an intelligent health care system online. The system is fed with various symptoms and the disease/illness associated with those systems. The system allows users to share their symptoms and issues. It then processes the user's symptoms to check for various illnesses that could be associated with it. Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with a patient's symptoms. In the doctor module when a doctor login to the system doctor can view his patient details and the report of that patient. Doctors can view details about the patient search and what the patient searched for according to their prediction. Doctor can view his personal details. Admin can add new disease details by specifying the type and symptoms of the disease into the database. Based on the name of the disease and symptom the data mining algorithm works. Admin can view various diseases and symptoms stored in the database. This system will provide proper guidance when the user specifies the symptoms of his illness.

CONTENTS

Chapter 1

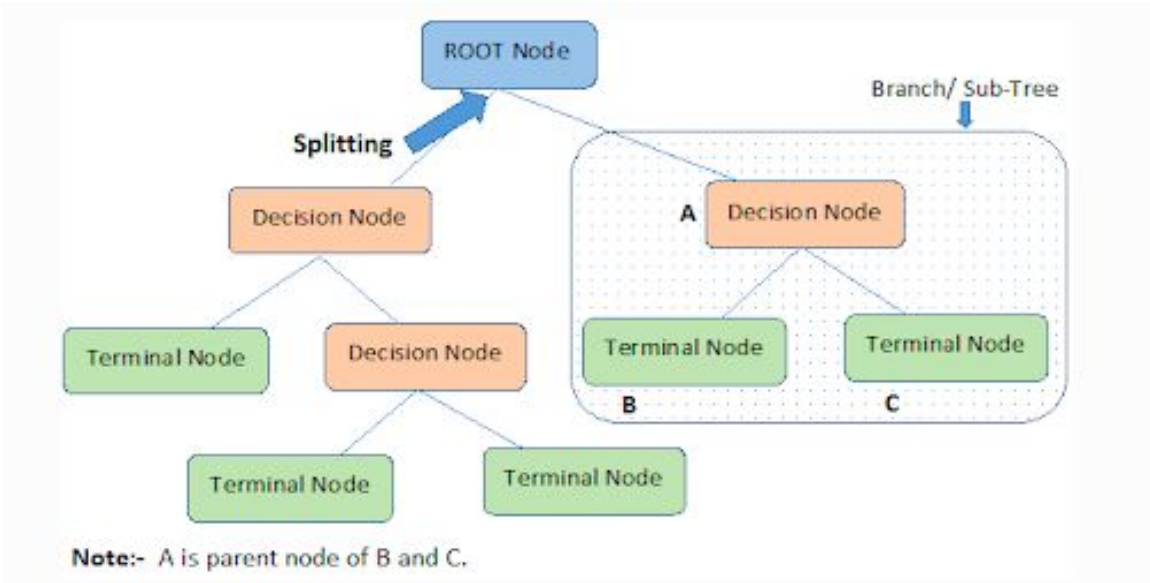
INTRODUCTION

1.1 Functional Structure

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.The goal of using a Decision Tree is to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

1.2 Important Terminology

- **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
- **Leaf / Terminal Node:** Nodes that do not split are called Leaf or Terminal nodes.
- **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
- **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
- **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.



If the dataset consists of N attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy.For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some criteria like Entropy, Information gain, Gini index, Gain Ratio, Reduction in Variance, Chi-Square.These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value (in case of information gain) is placed at the root. While using Information Gain as a criterion, we assume attributes to be categorical, and for the Gini index, attributes are assumed to be continuous.

1.3 Technology Stack

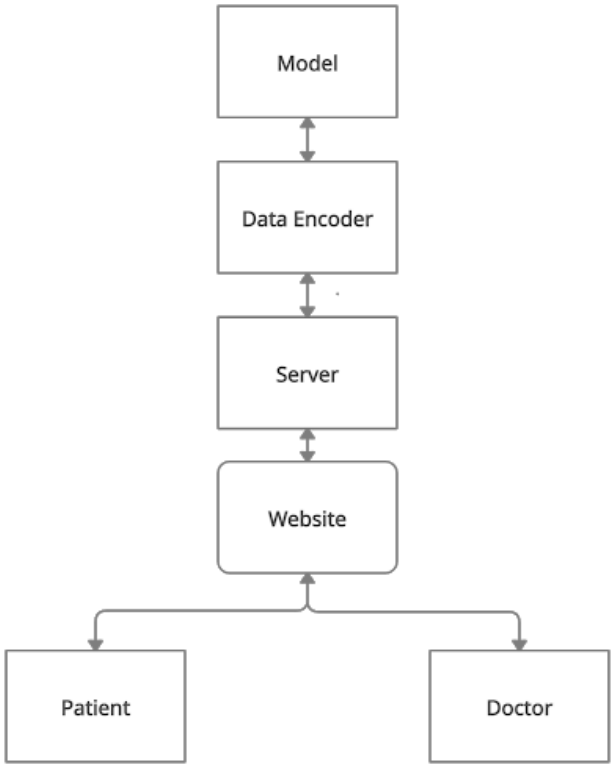
Data Science and Machine Learning libraries based on python were used in this project. The main tool used for coding was Jupyter Notebooks. Pandas and Numpy were used for data analysis. Streamlit was used to create a user interface for the web application. Finally Machine Learning models available in the Scikit Learn library were used for model training and evaluation primarily including decision tree and random forest algorithm.

Chapter 2

IMPLEMENTATION

2.1 Software Architecture

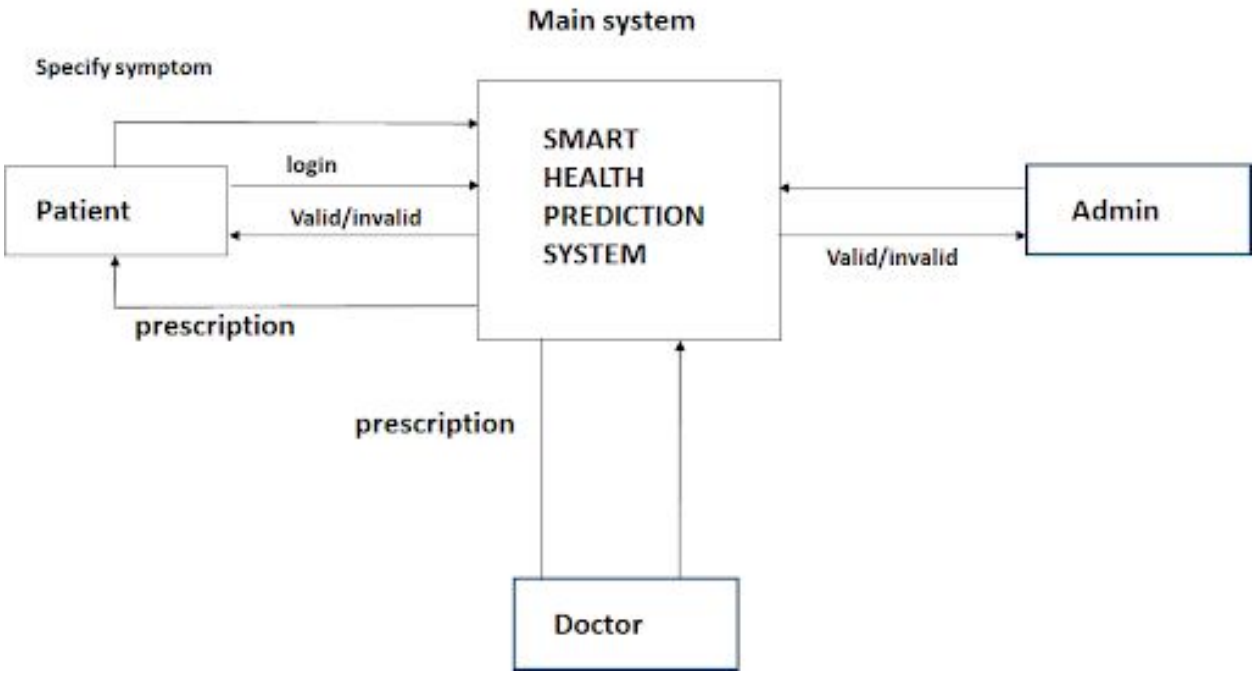
The software architecture follows an encoded path from model to hosted website on the user side, where the user has access to patient portal information and the doctors can see the database for all the previously evaluated symptoms and predictions, this inturn gives the doctor administrator rights to check and consult with the patient depending on the severity.



2.2 Design Architecture

It is an end user support and online consultation project. Here we propose a system that allows users to get instant guidance on their health issues through an online intelligent health care system. The system is fed with various symptoms and the disease/illness associated with those symptoms. The system will allow the users to share their symptoms and issues. It then processes the user's symptoms to check for various illnesses that could be associated with it. Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with a patient's symptoms. In the doctor module when a doctor logs in to the system doctor can view his patient details and the report.

Doctors can view details about the patient search and what patient searched for according to their prediction. Doctor can view his personal details. Admin can add new disease details by specifying the type and symptoms of the disease into the database. Based on the name of the disease and symptom the data mining algorithm works. Admin can view various diseases and symptoms stored in the database. This system will provide proper guidance when the user specifies the symptoms of his illness. In our project we propose a system that is user favorable to get guidance on health issues instantly through an online health care system.



2.3 Algorithm Code Base

This is the configuration code and the main driver code to create and save the model to be used for the final prediction.

```
# This file is for training and saving model files

# Import Dependencies
import yaml
from joblib import dump, load
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Trees Approach
from sklearn.tree import DecisionTreeClassifier
```

```
# Ensemble Approach
from sklearn.ensemble import RandomForestClassifier


class DiseasePrediction:
    # Initialize and Load the Config File
    def __init__(self, model_name=None):
        # Load Config File
        try:
            with open('./config.yaml', 'r') as f:
                self.config = yaml.safe_load(f)
        except Exception as e:
            print("Error reading Config file...")

        # Load Training Data
        self.train_features, self.train_labels, self.train_df =
self._load_train_dataset()
        # Load Test Data
        self.test_features, self.test_labels, self.test_df =
self._load_test_dataset()
        # Model Definition
        self.model_name = model_name
        # Model Save Path
        self.model_save_path = self.config['model_save_path']

    # Function to Load Train Dataset
    def _load_train_dataset(self):
        df_train = pd.read_csv(self.config['dataset']['training_data_path'])
        cols = df_train.columns
        cols = cols[:-2]
        train_features = df_train[cols]
        train_labels = df_train['prognosis']

        # Check for data sanity
        assert (len(train_features.iloc[0]) == 132)
        assert (len(train_labels) == train_features.shape[0])

        return train_features, train_labels, df_train

    # Function to Load Test Dataset
    def _load_test_dataset(self):
        df_test = pd.read_csv(self.config['dataset']['test_data_path'])
        cols = df_test.columns
        cols = cols[:-1]
        test_features = df_test[cols]
```

```
        test_labels = df_test['prognosis']

        # Check for data sanity
        assert (len(test_features.iloc[0]) == 132)
        assert (len(test_labels) == test_features.shape[0])

        return test_features, test_labels, df_test

    # Dataset Train Validation Split
    def _train_val_split(self):
        X_train, X_val, y_train, y_val = train_test_split(self.train_features,
self.train_labels,

test_size=self.config['dataset']['validation_size'],

random_state=self.config['random_state'])

        return X_train, y_train, X_val, y_val

    # Model Selection
    def select_model(self):
        if self.model_name == 'decision_tree':
            self.clf =
DecisionTreeClassifier(criterion=self.config['model']['decision_tree']['criteri
on'])
        elif self.model_name == 'random_forest':
            self.clf =
RandomForestClassifier(n_estimators=self.config['model']['random_forest']['n_es
timators'])

        return self.clf

    # ML Model
    def train_model(self):
        # Get the Data
        X_train, y_train, X_val, y_val = self._train_val_split()

        classifier = self.select_model()
        # Training the Model
        classifier = classifier.fit(X_train, y_train)
        # Trained Model Evaluation on Validation Dataset
        confidence = classifier.score(X_val, y_val)
        # Validation Data Prediction
        y_pred = classifier.predict(X_val)
        # Model Validation Accuracy
```

```
        accuracy = accuracy_score(y_val, y_pred)
        # Model Classification Report
        clf_report = classification_report(y_val, y_pred)

        print('\nTraining Accuracy: ', confidence)
        print('\nValidation Prediction: ', y_pred)
        print('\nValidation Accuracy: ', accuracy)
        print('\nClassification Report: \n', clf_report)

        # Save Trained Model
        dump(classifier, str(self.model_save_path + self.model_name +
".joblib"))


    # Function to Make Predictions on Test Data
    def make_prediction(self, saved_model_name=None, test_data=None):
        try:
            # Load Trained Model
            clf = load(str(self.model_save_path + saved_model_name +
".joblib"))
        except Exception as e:
            print("Model not found...")

        if test_data is not None:
            result = clf.predict(test_data)
            return result
        else:
            result = clf.predict(self.test_features)
        accuracy = accuracy_score(self.test_labels, result)
        clf_report = classification_report(self.test_labels, result)
        return accuracy, clf_report

if __name__ == "__main__":
    # Model Currently Training
    current_model_name = 'random_f'
    # Instantiate the Class
    dp = DiseasePrediction(model_name=current_model_name)
    # Train the Model
    dp.train_model()
    # Get Model Performance on Test Data
    test_accuracy, classification_report =
dp.make_prediction(saved_model_name=current_model_name)
    print("Model Test Accuracy: ", test_accuracy)
    print("Test Data Classification Report: \n", classification_report)
```

2.4 User Interface Code

The user interface is made using Streamlit, a fast lightweight data dashboard generating framework created using python.

```
# importing necessary modules
import streamlit as st
import joblib
import pandas as pd
import time

# loading model and list of symptoms
model = joblib.load("saved_model/random_f.joblib")
symptoms_list = ['itching', 'skin_rash', 'nodal_skin_eruptions',
'continuous_sneezing', 'shivering', 'chills', 'joint_pain', 'stomach_pain',
'acidity', 'ulcers_on_tongue', 'muscle_wasting', 'vomiting',
'burning_micturition', 'spotting_ urination', 'fatigue', 'weight_gain',
'anxiety', 'cold_hands_and_feets', 'mood_swings', 'weight_loss',
'restlessness', 'lethargy', 'patches_in_throat', 'irregular_sugar_level',
'cough', 'high_fever', 'sunken_eyes', 'breathlessness', 'sweating',
'dehydration', 'indigestion', 'headache', 'yellowish_skin', 'dark_urine',
'nausea', 'loss_of_appetite', 'pain_behind_the_eyes', 'back_pain',
'constipation', 'abdominal_pain', 'diarrhoea', 'mild_fever', 'yellow_urine',
'yellowing_of_eyes', 'acute_liver_failure', 'fluid_overload',
'swelling_of_stomach', 'swelled_lymph_nodes', 'malaise',
'blurred_and_distorted_vision', 'phlegm', 'throat_irritation',
'redness_of_eyes', 'sinus_pressure', 'runny_nose', 'congestion', 'chest_pain',
'weakness_in_limbs', 'fast_heart_rate', 'pain_during_bowel_movements',
'pain_in_anal_region', 'bloody_stool', 'irritation_in_anus', 'neck_pain',
'dizziness', 'cramps', 'bruising', 'obesity', 'swollen_legs',
'swollen_blood_vessels', 'puffy_face_and_eyes', 'enlarged_thyroid',
'brittle_nails', 'swollen_extremeties', 'excessive_hunger',
'extra_marital_contacts', 'drying_and_tingling_lips', 'slurred_speech',
'knee_pain', 'hip_joint_pain', 'muscle_weakness', 'stiff_neck',
'swelling_joints', 'movement_stiffness', 'spinning_movements',
'loss_of_balance', 'unsteadiness', 'weakness_of_one_body_side',
'loss_of_smell', 'bladder_discomfort', 'foul_smell_of urine',
'continuous_feel_of_urine', 'passage_of_gases', 'internal_itching',
'toxic_look_(typhos)', 'depression', 'irritability', 'muscle_pain',
'altered_sensorium', 'red_spots_over_body', 'belly_pain',
'abnormal_menstruation', 'dischromic _patches', 'watering_from_eyes',
'increased_appetite', 'polyuria', 'family_history', 'mucoid_sputum',
'rusty_sputum', 'lack_of_concentration', 'visual_disturbances',
'receiving_blood_transfusion', 'receiving_unsterile_injections', 'coma',
```

```
'stomach_bleeding', 'distention_of_abdomen', 'history_of_alcohol_consumption',
'fluid_overload.1', 'blood_in_sputum', 'prominent_veins_on_calf',
'palpitations', 'painful_walking', 'pus_filled_pimples', 'blackheads',
'scurring', 'skin_peeling', 'silver_like_dusting', 'small_dents_in_nails',
'inflammatory_nails', 'blister', 'red_sore_around_nose', 'yellow_crust_ooze']

# start of streamlit UI
st.title("Predico - The Smart health predictor")
st.header("Please enter your symptoms ")

symptoms = st.multiselect('Enter your symptoms so that we can get you a primary
diagnosis:',[*symptoms_list],key='symptoms')

# creating dataframe for accepting testing values
prediction_value = [0 for i in range(132)]
for sym in symptoms:
    index = symptoms_list.index(sym)
    # assigning encoded value to testing frame
    prediction_value[index] = 1

# convert list to Pandas dataframe and transpose it for model evaluation
query = pd.DataFrame(prediction_value).T
prediction = model.predict(query)

# evaluation and confirmation
if st.button("Evaluate"):
    with st.spinner('Predicting output...'):
        time.sleep(1)
        if symptoms:
            st.success("Prediction complete!")
            st.write("The diagnosis we have reached is: ")
            st.error(*prediction)
            st.write("Please consult your nearest health administrator soon,
take care! ")

        else:
            st.info("Please enter at least one symptom before clicking
evaluate)
```

Chapter 3

CONCLUSION

3.1 Results and Conclusions

Machine learning enabled individuals in tackling challenges and shortcomings that else would have been burdensome in a prudent manner. Machine learning tools aid to bring out insights on data to analyze patterns and to build models to make predictions. Having Machine learning methodologies being enforced in the health domain benefits for processing immense amounts of data beyond the scope of human ability, vivid predictions to be formulated with machine learning models and assistance for physicians for diagnosis in an efficacious manner. All those tedious and time consuming processes can be hastened to save both time and labour. Our project entitled in the name 'The Health Prediction system' provides assistance to determine the possible disease in reference to symptoms. However the challenges are still unsolved.Models are prone to overfitting that may end up in wrong predictions. Diagnosis cannot be done merely in light of symptoms, there exist various factors concerned about the patient that can lead to diseases. They include lifestyle,gender,hereditary etc. Advancements have to be brought in models to predict the disease based on factors other than symptoms which aids doctors to rely on these models for efficient disease predictions.

3.2 References

1. K.R.Lakshmi, Y.Nagesh and M.VeeraKrishna, "Performance comparison of three data mining techniques for predicting kidney disease survivability", International Journal of Advances in Engineering & Technology, Mar. 2014.
2. D. W. Bates, S. Saria, L. Ohno-Machado, A. Shah, and G. Escobar, "Big data in healthcare: using analytics to identify and manage high-risk and high-cost patients," Health Affairs, vol. 33, no. 7, pp. 1123–1131, 2014.
3. Disease prediction methodology: D. Dahiwade, G. Patle and E. Meshram, "Designing Disease Prediction Model Using Machine Learning Approach," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 1211-1215, doi: 10.1109/ICCMC.2019.8819782
4. Boshra Brahmi, Mirsaeid Hosseini Shirvani, "Prediction and Diagnosis of Heart Disease by Data Mining Techniques", Journals of Multidisciplinary Engineering Science and Technology, vol.2, 2 February 2015, pp.164- 168.
5. Streamlit official documentation: https://docs.streamlit.io/en/stable/
6. Decision tree and random forest exploration: https://www.khanacademy.org/computing/computer-science/informationtheory/info-theory/pi/decision-tree-exploration