ML Based Disease Predictor

Swati Shukla
Vanushri Rawat
Odkhuu Batmunkh

swati.shukla@sjsu.edu
vanushri.rawat@sjsu.edu
odkhuu.batmunkh@sjsu.edu

Advisor: Prof. Rakesh Ranjan

Department of Software Engineering, San Jose State University, California

Abstract Doctors have to research through lots of books and journals to get help with diagnosis of diseases. A single disease can occur for many reasons, lots of symptoms will be associated with a single disorder. There is a need to find a quick solution to cut out time for diagnosis. Machine learning(ML) can be the next solution. Prediction systems are rooted on logic based deductions and we have used Logistic Regression to train our dataset and label the data. We have used IBM Watson to implement Machine learning Models for 4 diseases: Heart Disease, Diabetes, Artery Disorder and Anxiety & Depression. The training model is based on real data set for heart disease and for the rest we have simulated datasets as close as possible to the real data. There is another use of the application for patients where they can ask questions and get first level of consultation from the chatbot. The bot will ask set of predefined questions related to symptoms, call the ML based Rest API and give a suggestion of consulting a doctor based on prediction.

We have also implemented our own ML to have comparisons with the IBM ML prediction.

Keywords ML (Machine Learning), Logistic Regression Algorithm, Chatbot, Incremental learning

1. INTRODUCTION

Predictive analysis or pattern recognition on huge datasets related to diseases, machine learning is the way to go. Machine Learning (ML) is the fastest rising arena in software industry and health informatics is of extreme challenge. The aim of Machine Learning is to develop algorithms which can learn and improve over time and can be used for disease and other health disorder predictions. Machine Learning practices are widely used in varied fields and health care industry has benefited a lot through machine learning prediction techniques. It offers a variety of prediction, analysis and decision supporting tools, targeted at improving patients' safety and quality of healthcare. With the need to predict diseases faster and more accurately based on past data, ML can help reduce the time to diagnose a disease with continuous learning. Machine Learning offers a wide variety of tools, techniques, and frameworks to address these challenges. We have used Watson ML models to apply binary classification based on Logistic Regression algorithm on our datasets. This project report provides a glimpse on the applications of Machine Learning Models highlighting its prominent role for health care industry and medical practitioners.

2. ARCHITECTURE

This web and mobile application is designed to provide users with analysis and prediction of diseases based on symptoms, test reports and previously trained data. At first, the featured data is used to train the model to predict the probability of diseased or non-diseased in the labeled data. This processed data is then used to provide data analysis and prediction.

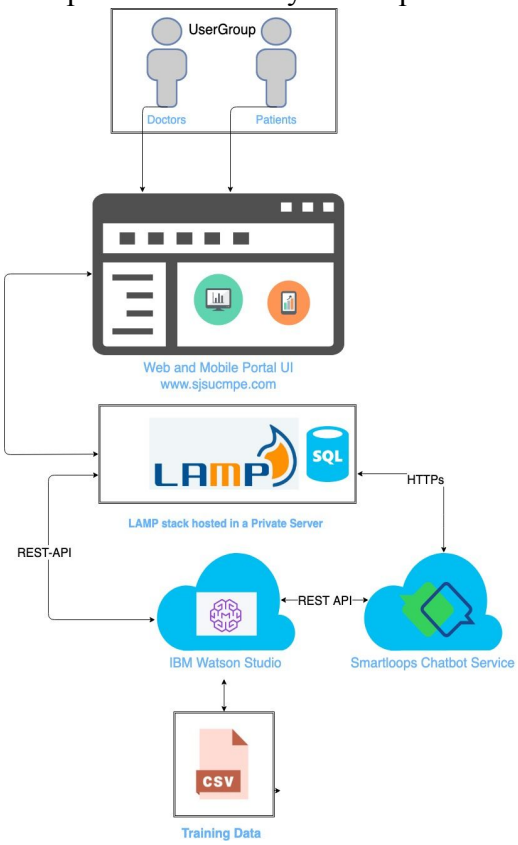


Figure: Architecture Diagram

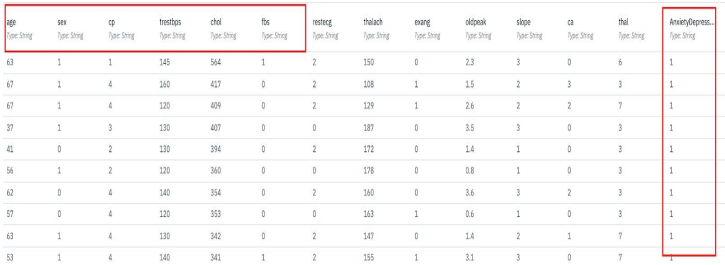
The disease prediction web/mobile application consists of the following layers:

2.1 DATA

For analysis and prediction, we used the data given on medical website[1]. It contains details related to heart diseases of patients from the year 1988.

We used one real dataset from the mentioned website and simulated the rest of them accordingly, close enough to the real data.

Below is the what the data looks like:



age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	AnxietyDepress...
63	1	1	145	564	1	2	150	0	2.3	3	0	6	1
67	1	4	160	417	0	2	108	1	1.5	2	3	3	1
67	1	4	120	409	0	2	129	1	2.6	2	2	7	1
37	1	3	130	407	0	0	187	0	3.5	3	0	3	1
41	0	2	130	394	0	2	172	0	1.4	1	0	3	1
56	1	2	120	360	0	0	178	0	0.8	1	0	3	1
62	0	4	140	394	0	2	160	0	3.6	3	2	3	1
57	0	4	120	353	0	0	163	1	0.6	1	0	3	1
63	1	4	130	342	0	2	147	0	1.4	2	1	7	1
53	1	4	140	341	1	2	155	1	3.1	3	0	7	1

The data used by us for training the Watson ML is labeled and has following features:

1) Age
2) Sex
3) Chest Pain
4) Blood Pressure
5) Cholesterol
6) Blood Sugar

The labeled column is the binary classified actual disease which needs to be predicted - in this case it is Anxiety & Depression.

2.2 ML TRAINING PROCESS

2.2.1 IBM WATSON

The labeled datasets were used to train the ML on IBM watson based on featured columns. Once the model was trained successfully they were deployed and ready for prediction.

Below are the various steps involved:

A. **Selecting an Algorithm** : First step of implementing an ML model is to select an algorithm that will give the desired results. For our project we have selected linear regression.
B. **Selecting the features** : Once the model is chosen and the dataset has been attached with it, selecting the features is the next step. This step actually lets the algorithm know on what basis the prediction needs to be made.

For our project we selected six features, as described before.

C. **Train Data Model** : Training the model means letting the machine make some patterns based on the features and connect it to the labeled column.
For our project we trained four data models.

D. **Deploying the Model** : This is the last step which makes the ML model accessible to the public on the IBM cloud. For our project we deployed our model on IBM cloud and accessed them through APIs.

2.2.2 Our own ML (written on Jupyter Notebook)

We used python based logistic regression algorithm for the ML model we designed in Jupyter Notebook[2] to solve the Binary Classification problem between on dependent variable and other independent variables. The outcomes are dichotomous and computes the probability of occurrence of events based of Maximum Likelihood estimation approach giving discrete output. Dependent variables follow Bern oulli's Distribution. Here is the head of dataset we used for our own model:

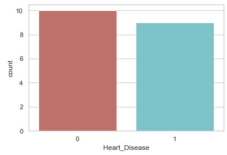
Out[144]:

	Age	Sex	Chest_Pain	Cholesterol	Heart_Disease
0	21	0	41.0	120	0
1	50	1	56.6	160	1
2	40	1	67.0	140	0
3	79	1	42.0	150	1
4	28	1	56.0	125	0

Age: Age Of Patient Sex: 0 means Female, 1 means Male Chest Pain: Range 41-67, 60+ is likely to have Heart Disease. Cholestrol: Range 120-160, 150+ is likely to have Heart Disease. Heart Disease: 0 = No Heart Disease 1 = Yes, have Heart Disease.

```
In [143]: data['Heart_Disease'].value_counts()
Out[143]: 0    10
          1     9
          Name: Heart_Disease, dtype: int64

In [144]: sns.countplot(x='Heart_Disease',data=data, palette='hls')
          plt.show()
```



We built a model on Scikit learning using pandas on a real heart disease medical dataset. The featured variables were Age, Sex, Chest Pain and Cholesterol just like the dataset in IBM Watson Training Model. The labeled or to be predicted column is Heart_Disease where '0' represents "No Heart Disease" and '1' represents "Heart Disease".

```
In [154]: feature_cols = ['Age', 'Sex', 'Chest_Pain', 'Cholesterol']

          X = data[feature_cols]
          y = data.Heart_Disease

In [169]: from sklearn.cross_validation import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
          print (X_train.shape, y_train.shape)
          print (X_test.shape, y_test.shape)

          (14, 4) (14,)
          (5, 4) (5,)

In [170]: logreg = LogisticRegression()
          # fit the model with data
          logreg.fit(X_train,y_train)
          #Prediction
          y_pred=logreg.predict(X_test)

          /Users/swatishukla/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default
          solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
            raise ValueError("multi_class should be either multinomial or "

In [171]: from sklearn import metrics
          cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
          cnf_matrix

Out[171]: array([[1, 0],
                 [2, 2]])
```

Then we will split the data into training set and test set. We broke the data into the ratio of 75:25 - 75% as training set and 25% as test set which is considered as good strategy.

2.3 ML ANALYTICS PROCESS

2.3.1 IBM WATSON

Integrated to work with external systems, Watson Machine Learning empowered our application to use the models easily through APIs. Once the models are deployed using them for prediction is the idea behind analytics process.

The models can be tested by specifying the featured columns in the test parameters and the analysis can be made. The values provided ideally be not existing in the dataset so that the prediction can be tested in real sense[3].

The output of the ML is in a JSON format which is converted into a graph by IBM and shown to the user. The response contains the prediction as well as the probability of certain kind of prediction.

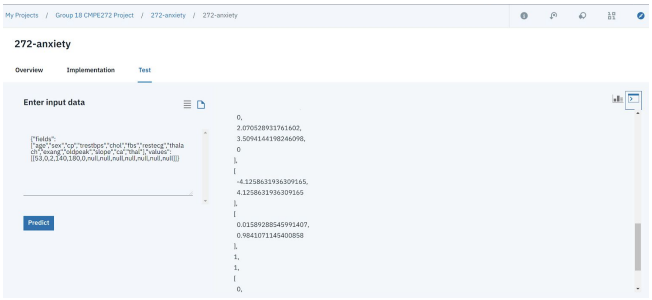


Figure : Test request and response in JSON format

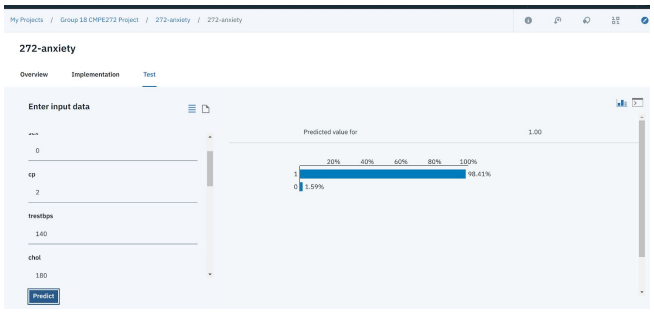


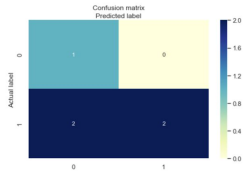
Figure : Test request and response in graphical format

2.3.3 Our own ML (written on Jupyter Notebook)

The model uses sigmoid function which is a 'S' shaped curve in which positive infinity value represents dependent variable is 1 and negative infinity value represents independent variable is 0.

We imported the Logistic Regression and created Logistic Regression classifier object. We then used Logistic Regression( ) function and fit the model and performed prediction using predict( ) function.

```
In [172]: class_names=[0,1] # name  of classes
          fig, ax = plt.subplots()
          tick_marks = np.arange(len(class_names))
          plt.xticks(tick_marks, class_names)
          plt.yticks(tick_marks, class_names)
          # create heatmap
          sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
          ax.xaxis.set_label_position("top")
          plt.tight_layout()
          plt.title('Confusion matrix', y=1.1)
          plt.ylabel('Actual label')
          plt.xlabel('Predicted label')
Out[172]: Text(0.5, 257.44, 'Predicted label')
```




```
In [173]: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
          print("Precision:",metrics.precision_score(y_test, y_pred))
          print("Recall:",metrics.recall_score(y_test, y_pred))
          Accuracy: 0.6
          Precision: 1.0
          Recall: 0.5
```

We created Confusion Matrix based on our predicted output which evaluates the performance of classification model. The output "correct or incorrect" is summed up classwise. In our case it was an array of "2*2" as the classification was binary which was as follows :

([[1, 0],
[2, 2]])

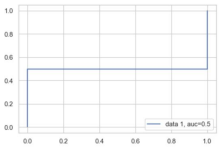
Diagonal 1 and 2 is correct and 0 and 2 is incorrect.

The accuracy of the classification rate of our ML Model is currently 0.6 (60%) which is considered good accuracy. We chose IBM Watson over our model as it's classification rate was more accurate.

The precision rate of our ML model and IBM Watson was same i.e. 1.0 (100%) which is considered excellent.

The recall of our ML model was just 0.5 (50%) whereas the recall of IBM watson was much more than it. Recall means if there are patients with disease in test set - the Logistic Regression model can identify it.

```
In [174]: y_pred_proba = logreg.predict_proba(X_test)[::,1]
          fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
          auc = metrics.roc_auc_score(y_test, y_pred_proba)
          plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
          plt.legend(loc=4)
          plt.show()
```



The ROC(Receiver Operating Characters) is a plot of the true positive rate against false positive rate which shows trade-off between sensitivity and specificity. In our model the AUC score for ROC is 1 which represents perfect classifier which was the same for Watson Model.

So while doing these comparisons we found Watson Model to be more precise and accurate than our own model that's why we preferred Watson over our ML model as the backbone of the project. We will improvise our own ML model for future usage for our application.

3. PROCESS FLOW

3.1 Technology integration - IBM Watson - Backend

As mentioned in section, project has 4 machine learning models deployed in IBM Watson Studio:

1. Anxiety prediction model
2. Artery disease prediction model

3. Diabetes disease prediction model
4. Heart disease prediction model

Each model was trained from 4 different datasets. After machine learning models have been trained and established, we have deployed our models. IBM Watson Studio enables models to be deployed in various different forms such as web service, batch prediction, and real-time streaming prediction. For our project, we have decided to deploy our models as a web-service so that we can reach our models through REST-API. In below figure please see how our 4 models have been deployed as a web-service.

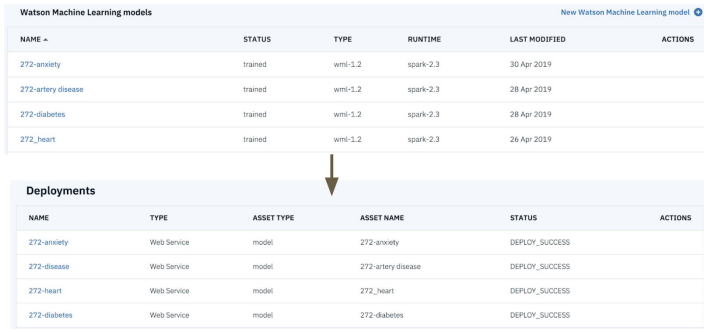


Figure: IBM Watson Machine Learning models deployed as a web-service

For our web-service deployments, we are passing user inputs in JSON format inside the “body” of the REST-API. Below is a sample “body” portion of a typical REST-API call against one of our models:

```
{"fields":["age","sex","cp","trestbps","chol",
"fbs","restecg","thalach","exang","oldpeak","s
lope","ca","thal"],"values":[[34,1,32,null,nul
l,null,null,null,null,null,null,null,null]]}
```

Figure: Sample JSON body of a REST-API call against ML model deployment

Once a model has been called, on IBM-watson dashboard, we can see that deployments return results in both graph and JSON formats. In our application we are receiving the JSON responses which we are interpreting through JavaScript to pick certain values from the response and display in graph form. Below please see a sample graph and JSON formats of a return of a prediction.

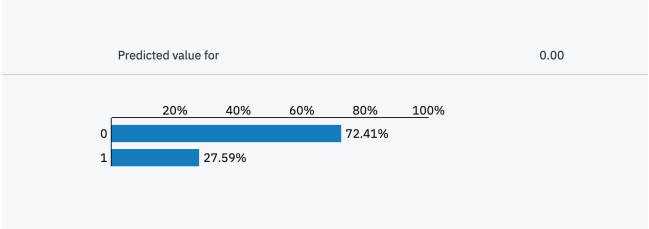


Figure: IBM Watson dashboard return of a ML prediction on a web-service deployment


```
{
  "fields": [
    "age",
    "sex",
    "cp",
    "trestbps",
    "chol",
    "fbs",
    "restecg",
    "thalach",
    "exang",
    "oldpeak",
    "slope",
    "ca",
    "thal",
    "features",
    "rawPrediction",
    "probability",
    "prediction",
    "nodeADP_class",
    "nodeADP_classes"
  ],
  "values": [
    [
      34,
      1,
      32,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      [
        3.717280069690115,
        2.091210597884843,
        33.43742467656244,
        7.414143903348816,
        4.846440990732382,
        0,
        1.007070013544375,
        0,
        -0.1286397939622407
      ],
      [
        21,
```

```
          8
        ],
        [
          0.7241379310344828,
          0.27586206896551724
        ],
        0,
        0,
        [
          0,
          1
        ]
      ]
    ]
}
```


Figure: Response JSON of a sample REST-API call against a ML model's web-service deployment

3.2 Technology integration - SmartLoops chatbot - Backend

SmartLoops provided us an open source platform to implement and integrate a chatbot into our project. The purpose of the chatbot was to provide an alternative method for more casual user input. In this case chatbot workflow was targeted for users such as patients. In our main workflow, the user input is done through a fill-in form with a predict button to call ML REST-APIs. However, in chatbot form, we are only collecting very basic data from a user (i.e. age, gender, chol-level, bp, etc) and calling our ML REST API against it. In below figure we can see how Smartloops allowed us to configure a REST-API call from a chat block.

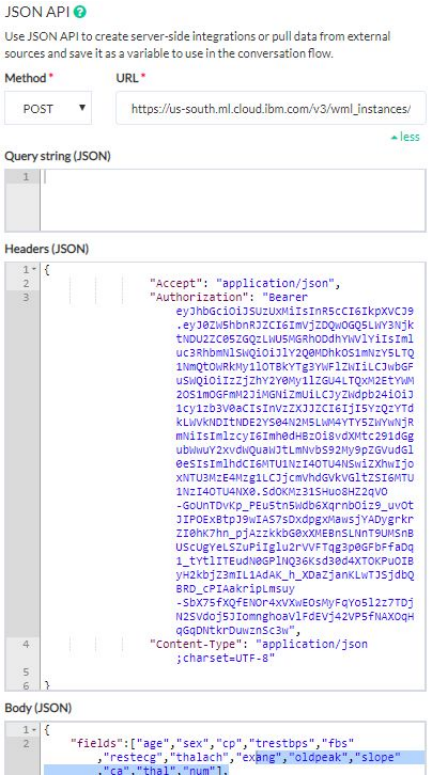


Figure: **SmartLoop integration to call IBM Watson MLs**

In the main form we are calling all four ML model deployments; however, in the chatbot we are only calling one ML against the Heart Disease prediction model. In Smartloop we were able to create simple chat blocks with conditional statements and user input validation for values such as gender, age, bp, chol-level, etc.

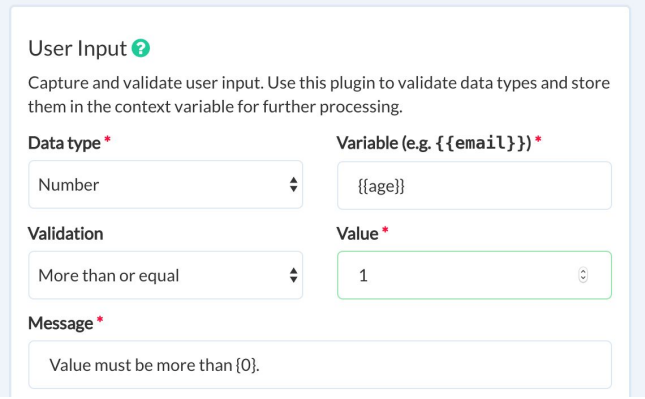


Figure: Sample user input validation in SmartLoops chatbot block for age input

We were able to integrate Smartloops chatbot box into our front end using code snippets that were generated by the platform to embed into our site:

```
<script>
  (function(d) {
    window['recime-bot'] =
{"id":"ef0118ee2ba8694b377fc6acdc33032b","titl
e":"Patient chatbot
272","apiKey":"9f8f92006d6d11e9ad0ce77a0c99f58
6","language":"en","primaryColor":"#ffabab","s
econdaryColor":"#eff4fa","icon":"comments","gr
eetingText":"Hi! How can we help
you?","greetingDelay":5,"playSound":true};
    var s = d.createElement('script');
    s.setAttribute('id',
'recime-chatbot-script');
    s.setAttribute('src',
'https://webchat.smartloop.ai/shim.js');
    s.setAttribute('async', 'async');

d.getElementsByTagName('body')[0].appendChild(
s);
  })(document);
</script>
```

Figure: Sample code snippet to embed Smartloops chatbot into our front end

In general, we have IBM Watson running as the backbone of our ML modeling and Smartloops running both chatbot logic and providing front-end for the chatbot.

3.3 Technology integration - Front-End LAMP stack

Since we have both ML and Chatbot logic running in the cloud, we have decided to implement our front-end simply in both forms: web and mobile.

We have hosted our website in a private server offsite on a Linux machine running LAMP stack. The site is hosted on an Apache server and has MySQL running as a database. Most of the front end such as user input forms and home page is written in JavaScript. Below please see sample Mobile and WebViews of our website.

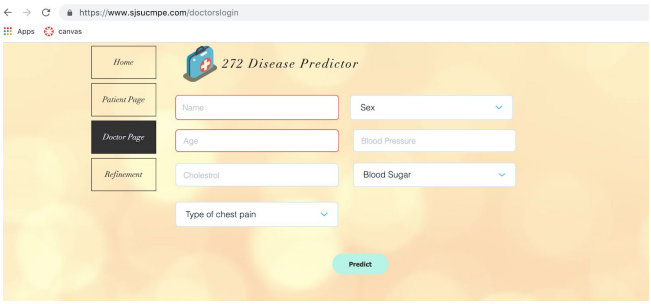


Figure: WebView of our website

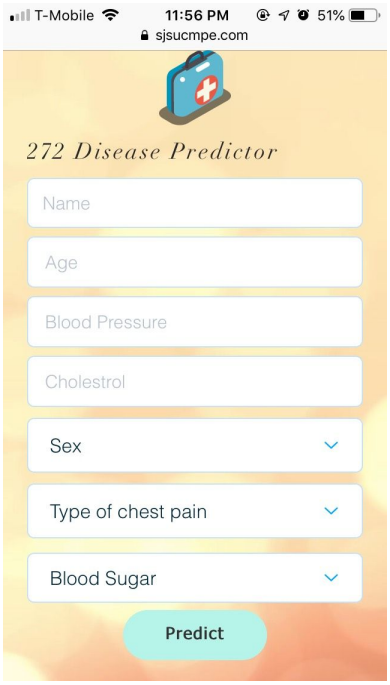


Figure: Mobile view of the Doctor Page

In general, in below figure please see general technology integration diagram between the ML models deployed in IBM Watson cloud and our front end hosted in an offsite private server.

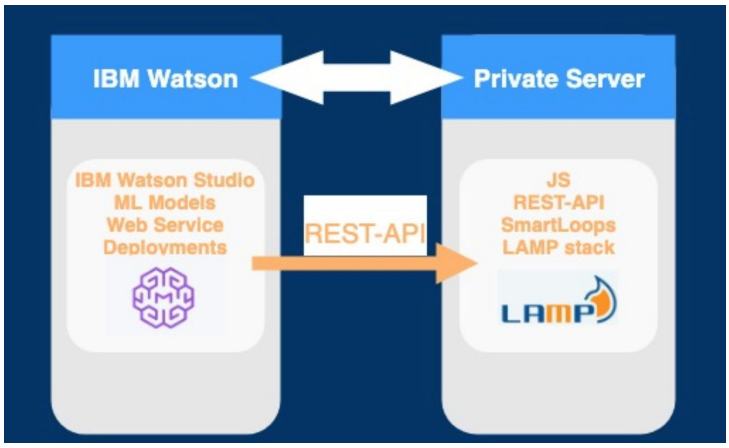


Figure: Overview technology integration diagram

3.3 Use-Cases

Doctor user:

- Doctors will login to the system
- Fill in symptoms and some test results
- Get the prediction based on past patient's data

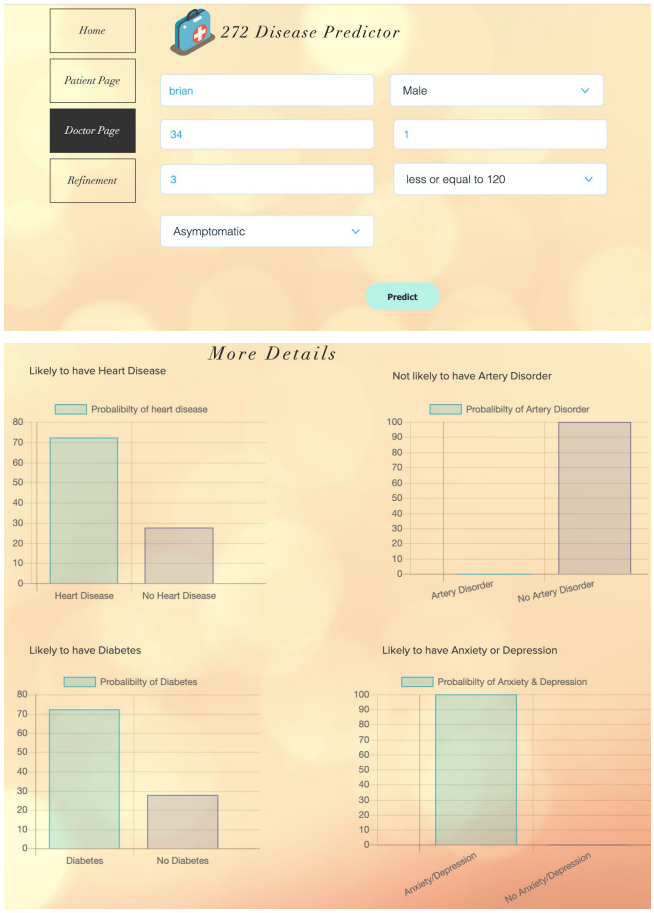


Figure: Sample prediction result for doctor's prediction

Doctor data refinement screen:

- Doctors will login to the system
- Give its own analysis of the patient and fill in the existing patient's diseases
- With the patient's consent add the new data to the dataset of ML

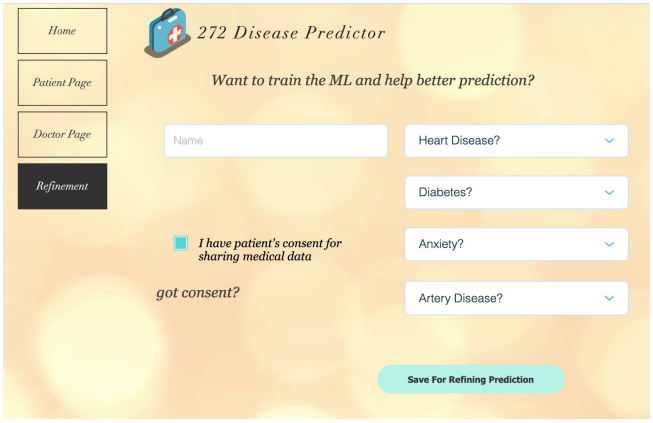


Figure: Data refinement page for Doctors input

Patient user:

- Patient will login to the system
- Get first level of consultation from the chatbot
- Chatbot will take some basic inputs and suggest how urgent it is for the patient to visit a doctor

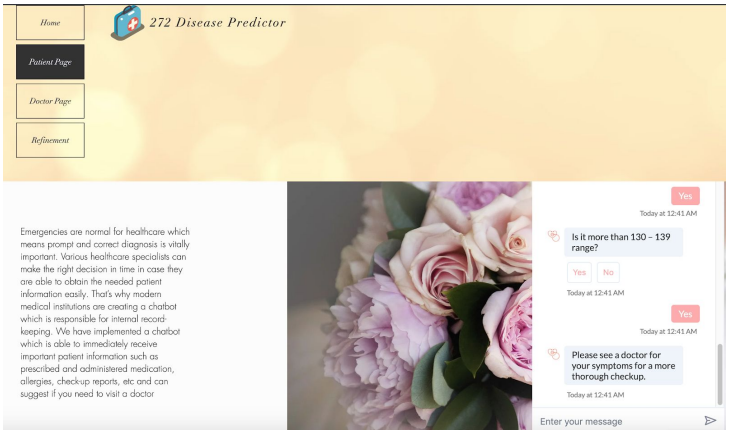


Figure: Sample Chatbot prediction for patient user

4. FUTURE ENHANCEMENT

Below are some of the enhancements that can be made to the solution :

4.1 Continuous Learning

The deployment is not a one-time event is our application, it needs manual intervention to refine the data and add new rows that the doctors will analyse. We will use IBM Watson Studio to retrain our model with new data and to do this we will use continuous learning system, which will provide automated monitoring of performance, retraining, and redeployment to ensure that the quality of the prediction is constantly increasing.

4.2 Disaster Recovery

With Professor Ranjan's feedback, we have realized that our system could be tailored to suit a disaster recovery scenario. In disaster situations, recovery experts will need to make quick decision. Having an ML based prediction system will help them make informed and researched decision in a short time when time is of value.

5. CONCLUSION

In conclusion, machine learning based disease prediction can be used in the medical field to make quick predictions and insights without spending too much time and resources. In real life scenarios, sometimes doctors are put in time constrained situations where they are forced to make quick decisions. In such emergency situations doctors usually make decisions based on their instincts and experience; however, having an ML based prediction system on hand is useful for doctors double check and proof their initial judgment.

6. ACKNOWLEDGEMENTS

We would like to thank Professor Ranjan for guiding us through the project and helping us pick a sensible and useful topic and also for providing us access to tools we used to complete the project such as IBM Watson Studio. Moreover, we would like to thank our classroom assistant Kavya Sahai for answering our technical questions.

7. PROJECT REPOSITORY

URL: https://www.sjsucmpe.com
GitHub Repo: https://github.com/SJSU272Spring2019/Project-Group-18

8. REFERENCES

[1]https://www.mldata.io/dataset-details/heart_disease/#customize_download
[2]https://www.analyticsvidhya.com/blog/2018/05/starters-guide-jupyter-notebook/
[3]https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-model-builder-tutorial-01-binary-classification-auto.html#step3