e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:11/November-2023 Impact Factor- 7.868 www.irjmets.com

MACHINE LEARNING-BASED DIAGNOSIS OF DISEASES

Ghanashyam Vagale*1, Durga Shankar Dalayi*2, Vedavathi Surada*3, Akkireddi Akash*4, Snehashish Srivastava*5, Vamsi Gedela*6

*1,2,3,4,5,6Dept., Of Computer Science & Engineering, RUAS, India.

ABSTRACT

The project includes the implementation of 3 linear models (Naïve Bayes), Support Vector Machine (Support Vector Machine) and K-Nearest Neighbors (KNet) models. These models are used to compare the performance of the three models on the diabetes and the heart disease datasets extracted from UCI's data repository. In addition to the algorithm comparison, each of these models has been incorporated into the prediction engine and is exposed over the API. The project also includes the implementation of a web platform that allows researchers and doctors to collaborate.

The results demonstrate that our prediction engine can not only detect the presence of a disease but also accurately predict it. It is possible to improve performance by using more complex deep learning techniques.

Keywords: Health Informatics, Machine Learning, Diagnosis, Diseases, Algorithms.

I. INTRODUCTION

The use of computing in medicine can be traced back to the early 1950's. The first applications of artificial intelligence (AI) in medicine can be found only in the 1970's through the development of expert systems (internist-I, mycin, oncosin). The use of AI in medicine was relatively limited in the United States before 1980. The first international conference on AI in medicine was held in Italy on September 13-14, 1985. The aim of the conference was to create a research community in medicine. In 1986, the society for Artificial intelligence in medicine (Society for AI in Medicine) was established, which hosts biennial international conferences. One of the challenges of using AI in medicine is the lack of data. The solution to this problem can be found in the development of electronic medical records (EMRs) in the late 1960's and early 1970's. In the 1980's, the main research theme was "Knowledge engineering", but in the 1990's and 2000's the main research topic was "Machine learning and data mining".

Types of Diabetes:

Type 1 diabetes is caused by a pancreas failure to deliver enough insulin. This type is called "Insulin Dependent Polygenic Disease mellitus" or "Juvenile Diabetes". The cause is unknown. The type 1 polygenic disease is found in children under the age of 20. People suffer from type 1 diabetes for the rest of their lives and rely on insulin injections. The type 1 diabetic patients often have to follow their doctor's recommendations for exercise and diet.

Type 2 diabetes begins with hypoglycemic agent resistance, where cells fail to effectively respond to the hypoglycemic agents. The disease develops due to a lack of hypoglycemia agent that additionally builds up. The type 2 diabetes is called "Non-Insulindependent Polygenic Disease Mellitus". Extreme weight is the common cause of type 2 diabetes. The number of people suffering from type 2 diabetes will increase by 3% by 2025. Diabetes mellitus is concentrated in rural areas compared to urban areas. Prehypertension and obesity are associated with type 2 diabetes. According to a United Nations agency, an individual has traditional vital signs.

II. METHODOLOGY

Agile: Agile is a project management methodology that involves segmenting a project into multiple phases, engaging stakeholders continuously, and iterating and improving the process continuously. Throughout the project's software development life cycle, it encourages ongoing iterations of testing and development. Testing and development are done simultaneously.

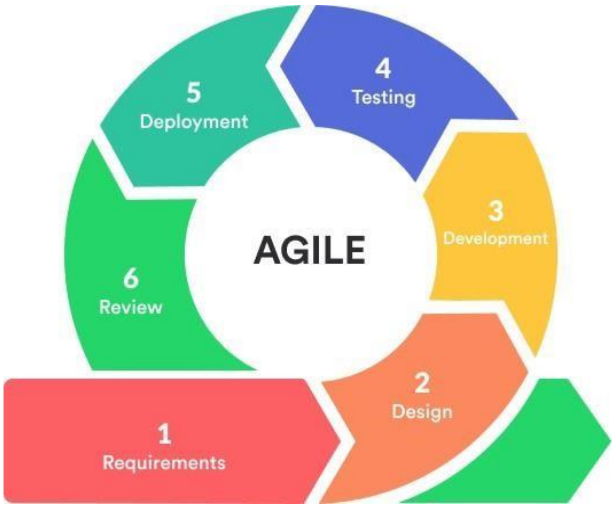


Fig 1: Agile Methodology

Scrum: SCRUM is an agile development approach that focuses on task management in a collaborative development environment. Scrum promotes self-organization and experience-based learning in teams as they tackle problems.

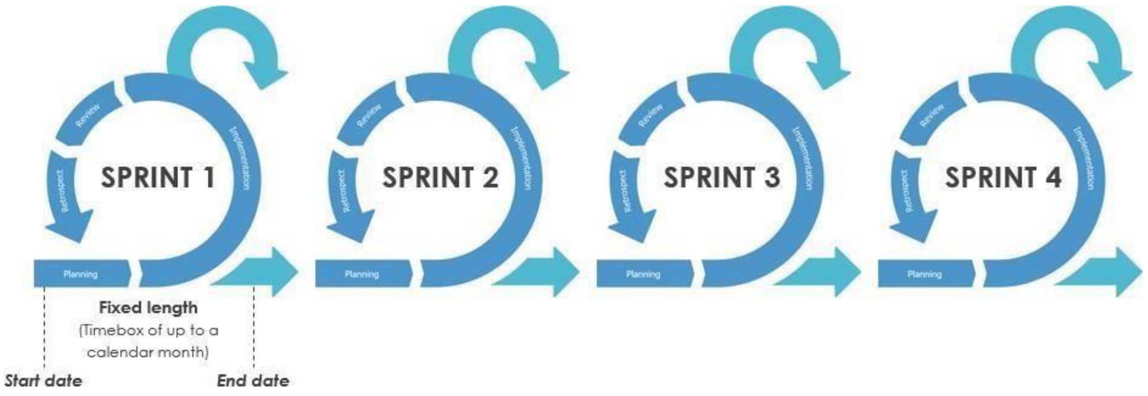


Fig 2: Scrum Methodology

Principal Artefacts:

- The Product Backlog, which is managed by the Product Owner or Product Manager, is the comprehensive list of tasks that require completion.
- The list of items, user stories, or bug fixes that the development team has decided to implement during the current sprint cycle is called the Sprint Backlog.
- The useable outcome of a sprint is called an increment, sometimes known as a sprint goal.

PROPOSED SYSTEM

1) The online application uses HTML forms to collect user input.
2) To send and receive data to the APIs, the Web application uses HTTP modules.
3) User input will be sent to the APIs as a JSON object (key-value pair).
4) The local system's flask file hosting consumes learned KNN, SVM, and Naïve Bayes models in the form of pickle files.
5) The user input JSON object is passed to the trained models.
6) The outcome of the prediction is delivered in response to the API calls. Heart disease and diabetes were predicted using trained KNN, SVM, and Naïve Bayes models. The models were trained on pre-processed standard datasets after the dataset was normalised using Standard Scalar.

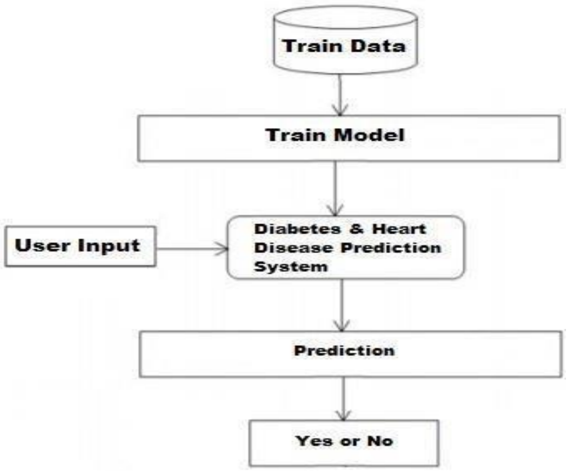
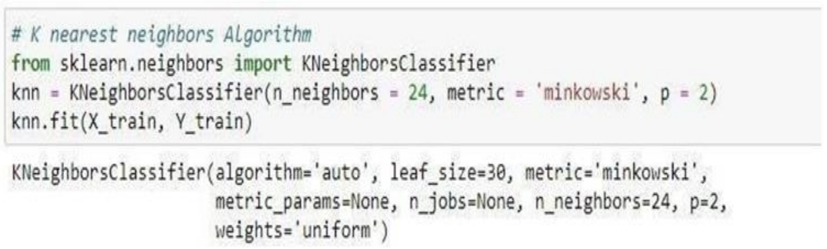
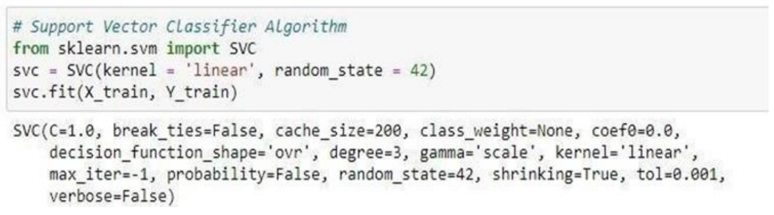


Fig 3: Flowchart of the method



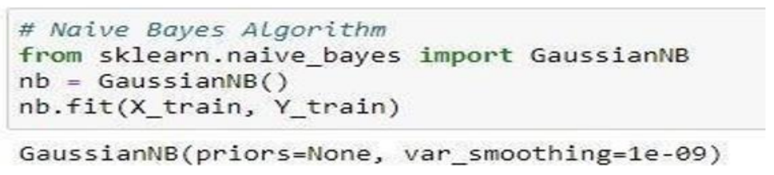
```
# K nearest neighbors Algorithm
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 24, metric = 'minkowski', p = 2)
knn.fit(X_train, Y_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=24, p=2,
                     weights='uniform')
```

Fig -4: Training of KNN Model



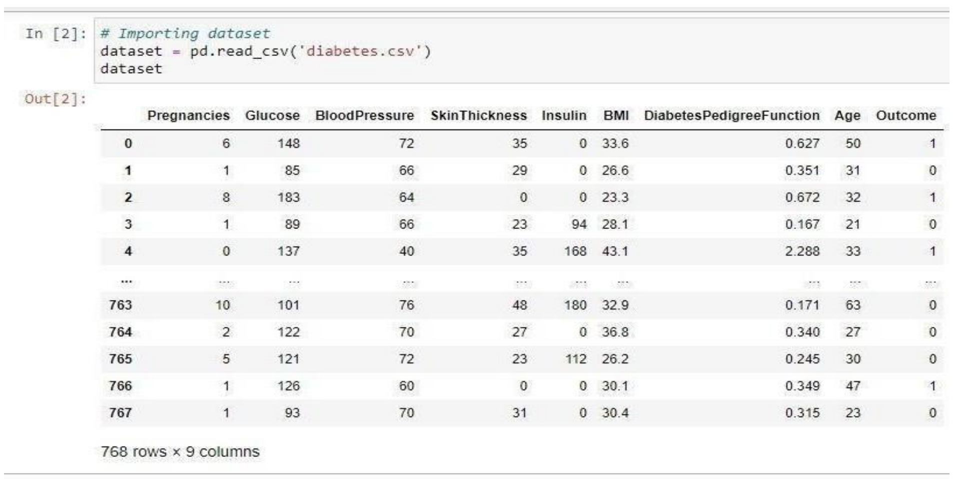
```
# Support Vector Classifier Algorithm
from sklearn.svm import SVC
svc = SVC(kernel = 'linear', random_state = 42)
svc.fit(X_train, Y_train)

SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=42, shrinking=True, tol=0.001,
    verbose=False)
```

Fig -5: Training of SVM Model



```
# Naive Bayes Algorithm
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, Y_train)

GaussianNB(priors=None, var_smoothing=1e-09)
```

Fig -6: Training of Naïve Bayes Model

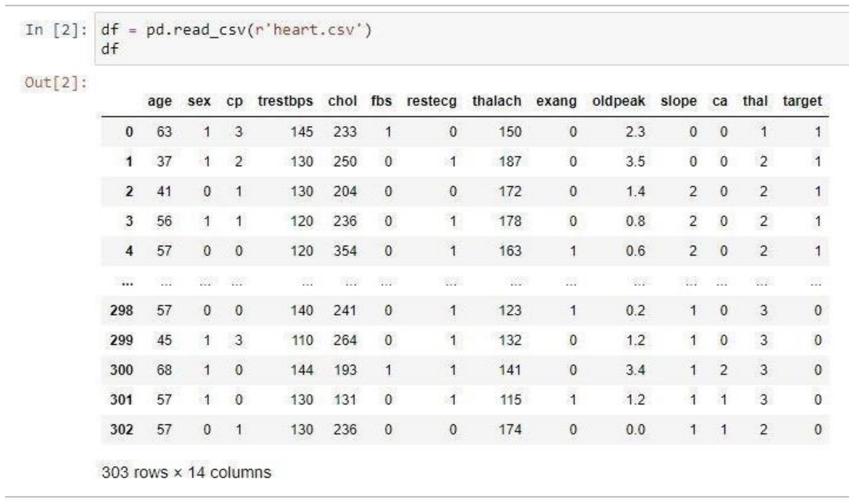


```
In [2]: # Importing dataset
        dataset = pd.read_csv('diabetes.csv')
        dataset
Out[2]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Fig 7: Pre-processed Diabetes Dataset



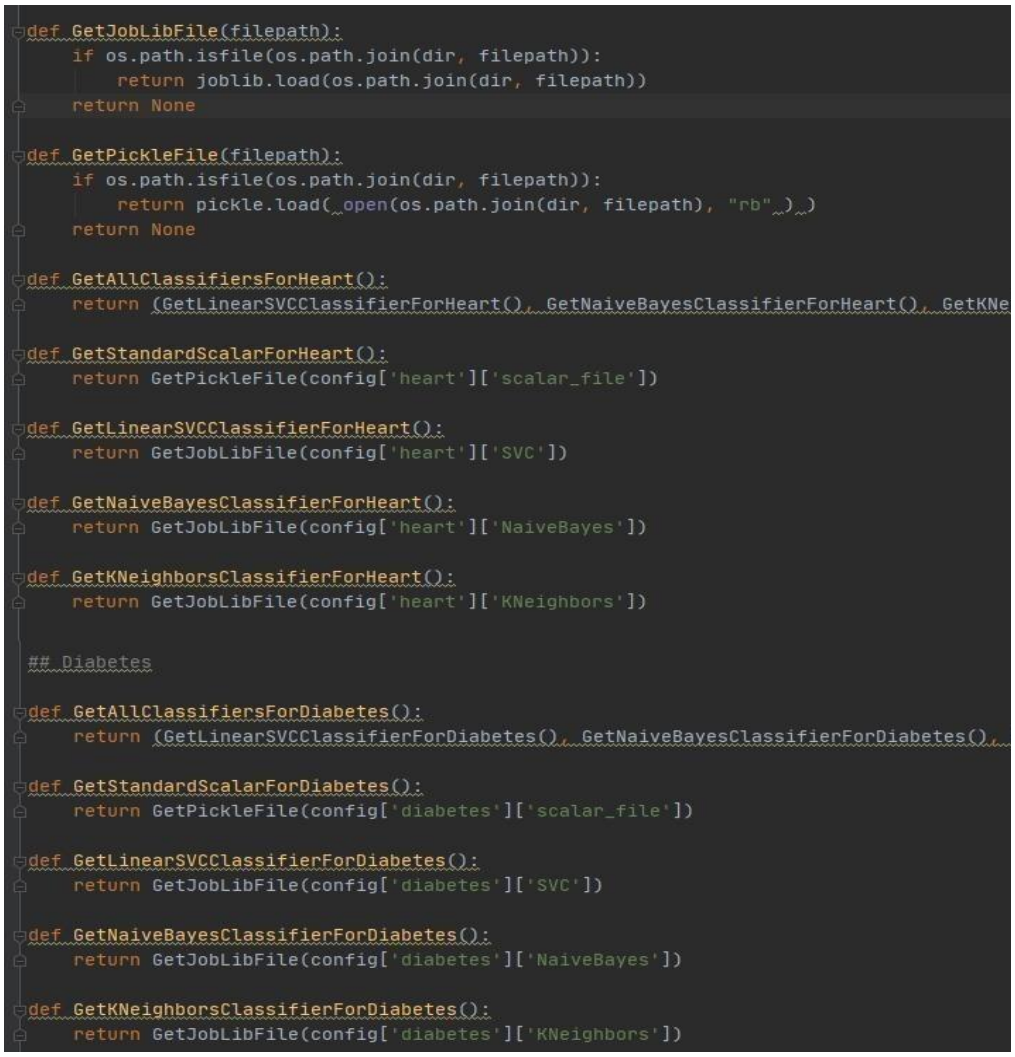
```
In [2]: df = pd.read_csv(r'heart.csv')
        df

Out[2]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Fig 8: Pre-processed Heart Disease Dataset

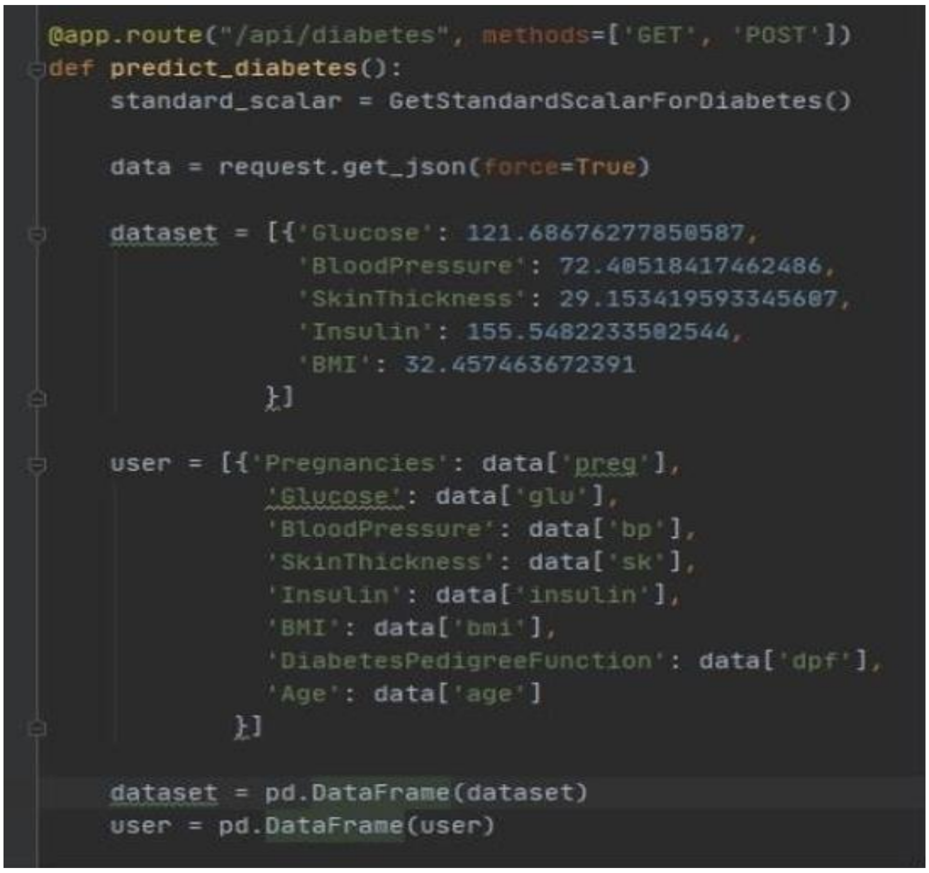
After the models were trained, they were extracted as pickle files and saved locally. The flask framework then uses this local storage to call the trained model when it receives user input.



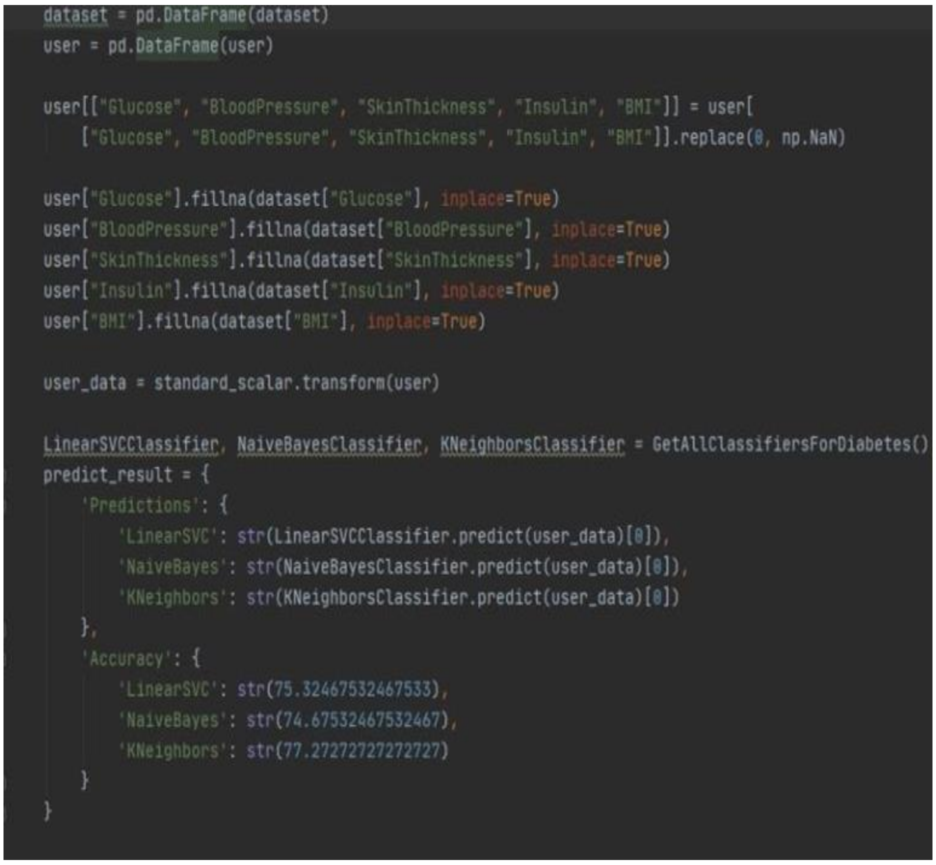
```
def GetJobLibFile(filepath):
    if os.path.isfile(os.path.join(dir, filepath)):
        return joblib.load(os.path.join(dir, filepath))
    return None

def GetPickleFile(filepath):
    if os.path.isfile(os.path.join(dir, filepath)):
        return pickle.load( open(os.path.join(dir, filepath), "rb" ) )
    return None

def GetAllClassifiersForHeart():
    return (GetLinearSVCClassifierForHeart(), GetNaiveBayesClassifierForHeart(), GetKNe

def GetStandardScalarForHeart():
    return GetPickleFile(config['heart']['scalar_file'])

def GetLinearSVCClassifierForHeart():
    return GetJobLibFile(config['heart']['SVC'])

def GetNaiveBayesClassifierForHeart():
    return GetJobLibFile(config['heart']['NaiveBayes'])

def GetKNeighborsClassifierForHeart():
    return GetJobLibFile(config['heart']['KNeighbors'])

## Diabetes

def GetAllClassifiersForDiabetes():
    return (GetLinearSVCClassifierForDiabetes(), GetNaiveBayesClassifierForDiabetes(),

def GetStandardScalarForDiabetes():
    return GetPickleFile(config['diabetes']['scalar_file'])

def GetLinearSVCClassifierForDiabetes():
    return GetJobLibFile(config['diabetes']['SVC'])

def GetNaiveBayesClassifierForDiabetes():
    return GetJobLibFile(config['diabetes']['NaiveBayes'])

def GetKNeighborsClassifierForDiabetes():
    return GetJobLibFile(config['diabetes']['KNeighbors'])
```

Fig 9: Returning trained models to the API's on function calls

Using the Flask framework, two APIs were created, one for diabetes prediction and the other for heart disease prediction. These APIs were hosted locally and will be used by the prediction engine's front end.



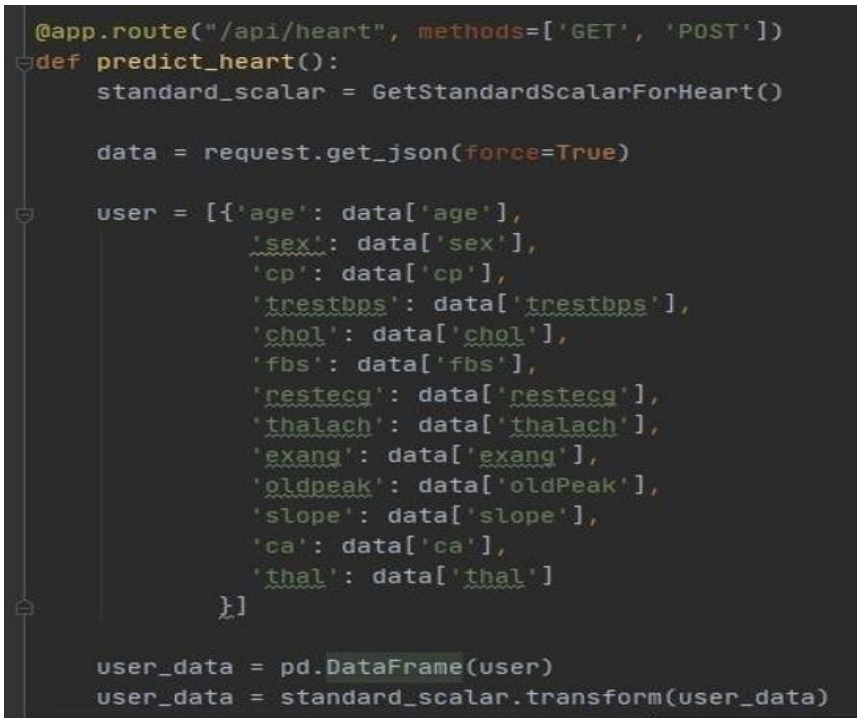
```
@app.route("/api/diabetes", methods=['GET', 'POST'])
def predict_diabetes():
    standard_scalar = GetStandardScalarForDiabetes()

    data = request.get_json(force=True)

    dataset = [{'Glucose': 121.68676277850587,
                'BloodPressure': 72.40518417462486,
                'SkinThickness': 29.153419593345687,
                'Insulin': 155.5482233502544,
                'BMI': 32.457463672391
              }]

    user = [{'Pregnancies': data['preg'],
             'Glucose': data['glu'],
             'BloodPressure': data['bp'],
             'SkinThickness': data['sk'],
             'Insulin': data['insulin'],
             'BMI': data['bmi'],
             'DiabetesPedigreeFunction': data['dpf'],
             'Age': data['age']
           }]

    dataset = pd.DataFrame(dataset)
    user = pd.DataFrame(user)
```



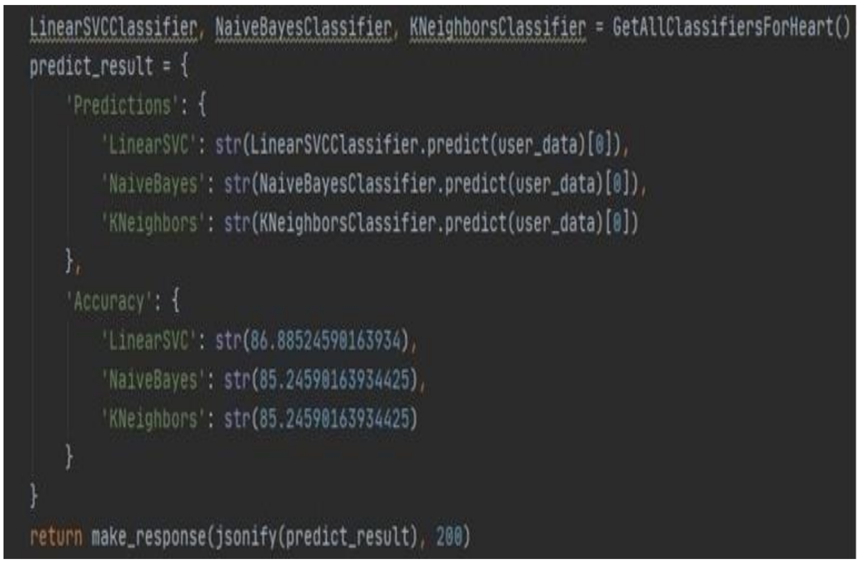
```
dataset = pd.DataFrame(dataset)
user = pd.DataFrame(user)

user[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = user[
    ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]].replace(0, np.NaN)

user["Glucose"].fillna(dataset["Glucose"], inplace=True)
user["BloodPressure"].fillna(dataset["BloodPressure"], inplace=True)
user["SkinThickness"].fillna(dataset["SkinThickness"], inplace=True)
user["Insulin"].fillna(dataset["Insulin"], inplace=True)
user["BMI"].fillna(dataset["BMI"], inplace=True)

user_data = standard_scalar.transform(user)

LinearSVCClassifier, NaiveBayesClassifier, KNeighborsClassifier = GetAllClassifiersForDiabetes()
predict_result = {
    'Predictions': {
        'LinearSVC': str(LinearSVCClassifier.predict(user_data)[0]),
        'NaiveBayes': str(NaiveBayesClassifier.predict(user_data)[0]),
        'KNeighbors': str(KNeighborsClassifier.predict(user_data)[0])
    },
    'Accuracy': {
        'LinearSVC': str(75.32467532467533),
        'NaiveBayes': str(74.67532467532467),
        'KNeighbors': str(77.27272727272727)
    }
}
```

Fig 10: API to predict Diabetes

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:11/November-2023 Impact Factor- 7.868 www.irjmets.com



```
@app.route("/api/heart", methods=['GET', 'POST'])
def predict_heart():
    standard_scalar = GetStandardScalarForHeart()

    data = request.get_json(force=True)

    user = [{'age': data['age'],
             'sex': data['sex'],
             'cp': data['cp'],
             'trestbps': data['trestbps'],
             'chol': data['chol'],
             'fbs': data['fbs'],
             'restecg': data['restecg'],
             'thalach': data['thalach'],
             'exang': data['exang'],
             'oldpeak': data['oldPeak'],
             'slope': data['slope'],
             'ca': data['ca'],
             'thal': data['thal']
           }]

    user_data = pd.DataFrame(user)
    user_data = standard_scalar.transform(user_data)
```



```
LinearSVCClassifier, NaiveBayesClassifier, KNeighborsClassifier = GetAllClassifiersForHeart()
predict_result = {
    'Predictions': {
        'LinearSVC': str(LinearSVCClassifier.predict(user_data)[0]),
        'NaiveBayes': str(NaiveBayesClassifier.predict(user_data)[0]),
        'KNeighbors': str(KNeighborsClassifier.predict(user_data)[0])
    },
    'Accuracy': {
        'LinearSVC': str(86.88524590163934),
        'NaiveBayes': str(85.24590163934425),
        'KNeighbors': str(85.24590163934425)
    }
}
return make_response(jsonify(predict_result), 200)
```

Fig 11: API to predict Heart Disease

III. MODELING AND ANALYSIS

Use Cases

Use cases were developed to better understand the requirements of the system, even though feature-driven development and agile development do not recommend using them. The domain areas are used to package the use cases. This section contains a detailed description of only a few of the use cases.

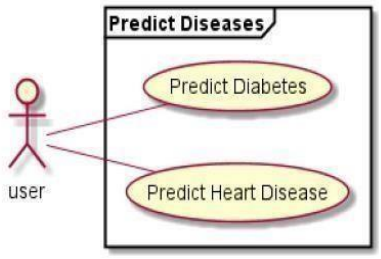


Fig 12: API to predict Heart Disease

Initial Engine Design

The prediction engine's initial algorithm was designed in Microsoft Visio before any code was written. The figure below shows the original design.

Every classification algorithm uses the same version of the algorithm below. Furthermore, 10-fold cross validation is used to calculate the accuracy score.

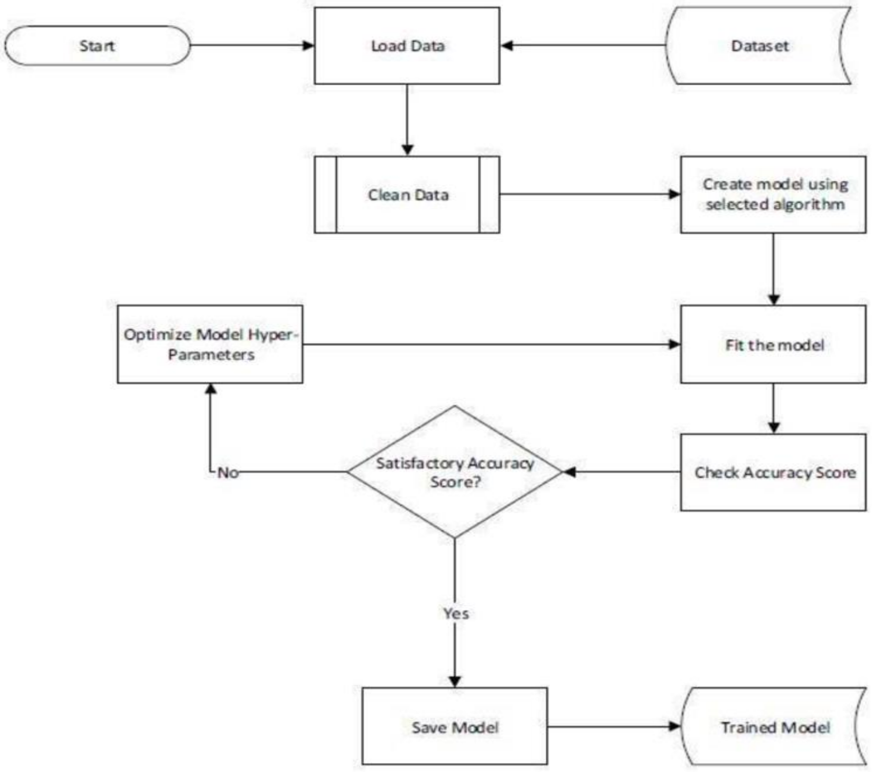

Start
Load Data
Dataset
Clean Data
Create model using selected algorithm
Optimize Model Hyper-Parameters
Fit the model
No
Satisfactory Accuracy Score?
Check Accuracy Score
Yes
Save Model
Trained Model

Fig 13: Training Model Process

The description of use cases for predicting diseases shown in Figure 12 is given below.:

Case Number: 1

Title: Predict Heart Disease

Description: The user submits the form with the values obtained after performing various medical tests. Upon submission, the user receives message indicating whether they have a risk of developing heart disease.

Actors: User

Priority: Essential

Postconditions: Patient successfully receives the prediction results.

Main Success Scenario:

1. The user performs all the tests and enters valid test results
2. The system performs prediction using the prediction engine API.
3. The user receives the prediction results.

Use Case Number: 2

Title: Predict Diabetes

Description: The user submits the form with the values obtained after performing various medical tests. Upon submission, the user receives a message indicating whether they have a risk of developing diabetes.

Actors: User Priority:

Essential

Postconditions: Patient successfully receives the prediction results. Main

1. The user performs all the tests and enters valid test results.

2. The system performs prediction using the prediction engine API.
3. The user receives the prediction results

Alternative Scenario:

If the user enters invalid value for any of the provided form fields: the system will display an error message.

If the system is unable to communicate with the prediction engine API: the system will display an error message.

IV. RESULTS AND DISCUSSION

The Prediction Engine provides an optimal performance with the right dataset and efficient training of the classifier models considering all aspects and a lot of learning from the previous experiences. The implemented Prediction Engine is capable of predicting the presence of Diabetes with an accuracy of:

• Linear SVM: 75.32467532467533

• Naïve Bayes: 74.67532467532467

• KNN: 77.27272727272727

And the presence of Heart Disease with an accuracy of:

• Linear SVM: 86.88524590163934

• Naïve Bayes: 85.24590163934425

• KNN: 85.24590163934425

Table 1. Comparison of algorithms for prediction of Diabetes and Heart Disease:

	Linear SVM	Naïve Bayes	KNN
Diabetes	75.32467532467533	74.67532467532467	77.27272727272727
Heart Diseases	86.88524590163934	85.24590163934425	85.24590163934425

The prediction engine's front-end was created using the Angular Framework, which includes HTML, CSS, and Typescript. The Angular Framework's HTTP Module was imported in order to handle the API calls and responses.

V. CONCLUSION

We created a prediction engine that allows the user to determine if they are at risk of heart disease or diabetes. In order to communicate with the prediction engine, the user must fill out a form with the parameter set that is used as an input for the trained models. When compared to other state-of-the-art methods, the prediction engine offers the best performance. Three algorithms are used by the prediction engine to determine whether a disease is present: Support Vector Machine (SVM), K-Nearest Neighbours (KNN), and Naïve Bayes. The following three algorithms were selected:

- If there is a lot of training data, they work well.
- All three of these algorithms can accept a single dataset as input with little to no changes.
- You can use a common scalar to normalise the input that these three algorithms receive.

VI. REFERENCES

[1] Priyanka Sonar, Prof. K. JayaMalini, "Diabetes Prediction Using Different Machine Learning Approaches", Proceedings of the Third International Conference on Computing Methodologies and Communication (ICCMC 2020) IEEE Xplore Part Number: CFP19K25-ART; ISBN: 978-1-5386-7808-4

[2] Deeraj Shetty, Kishor Rit, Sohail Shaikh, Nikita Patil, "Diabetes Disease Prediction Using Data Mining", 2021 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)

[3] Zhilbert Tafa, Nerxhivane Pervetica, Bertran Karahoda, "An Intelligent System for Diabetes Prediction", 4thMediterranean Conference on Embedded Computing MECO – 2020 Budva, Montenegro

[4] Dangare, Chaitrali & Apte, Sulbha. (2012). Improved Study of Heart Disease Prediction System using

Data Mining Classification Techniques. International Journal of Computer Applications. 47. 44-48. 10.5120/7228-0076.

[5] Shadab Adam Pattekari and Asma Parveen "PREDICTION SYSTEM FOR HEART DISEASE USING NAIVE BAYES" International Journal of Advanced Computer and Mathematical Sciences ISSN 2230-9624. Vol 3, Issue 3, 2012, Pages 290-294

[6] P. Groves, B. Kayyali, D. Knott, and S. van Kuiken, The'Big Data 'Revolution in Healthcare accelerating Value and Innovation. USA: Center for US Health System Reform Business Technology Office, 2016.

[7] M. Chen, Y. Hao, K. Hwang, L. Wang and L. Wang, "Disease Prediction by Machine Learning Over Big Data From Healthcare Communities," in IEEE Access, vol. 5, pp. 8869-8879, 2017