1. Introduction

Diabetes is a chronic disease that is a common threat to the health safety of western people [1]. Characterized by elevated glucose levels in the blood over extended periods of time. Easy, noninvasive predictions of diabetes will help prevent poor health outcomes and aid doctors in patients in providing affordable, and accessible healthcare.

The Pima Indian Diabetes Dataset[2] was created to help be able to predict that was created by the National Institute of Diabetes and Digestive and Kidney Diseases and published in 1988. The original group was able to predict the onset of diabetes at approximately 76% using neural networks. This has been a toy dataset for several years afterwards to test new techniques and methods. Diabetes is a serios disease and causes progressive damage. Early diagnostics help to prevent damage and help the medical course of the disease.

The indicators that were selected allowed minimally invasive tests which are useful for areas that lack medical infrastructure. At this time A1C testing was not available, so simple tests were available. The dataset is still useful for helping identify medical algorithms.

2. Related Work

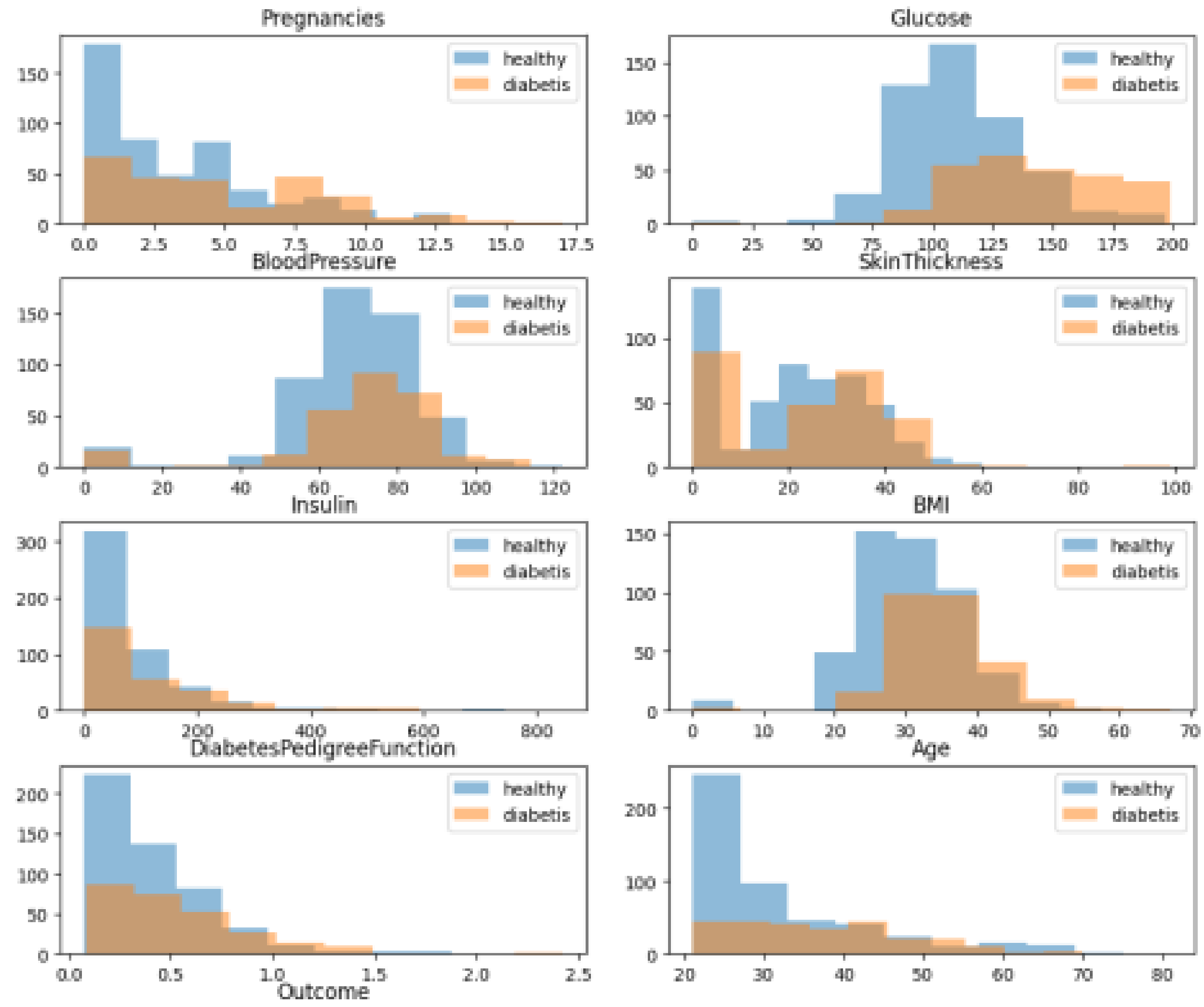
The Pima Indian Diabetes Dataset has been explored in many ways. The original dataset was explored with a neural network [2]. Neural Networks, Naive Bayes, Support Vector Machines, k nearest Neighbors, Random Forest, and Boost techniques [3,4]. As of this writing, almost 2709 [6] people have attempted to create a classifier on Kaggle to pull slightly higher accuracies for the dataset. It has become a successful toy dataset to use a learning metric for classifier.

3. Proposed Methodology

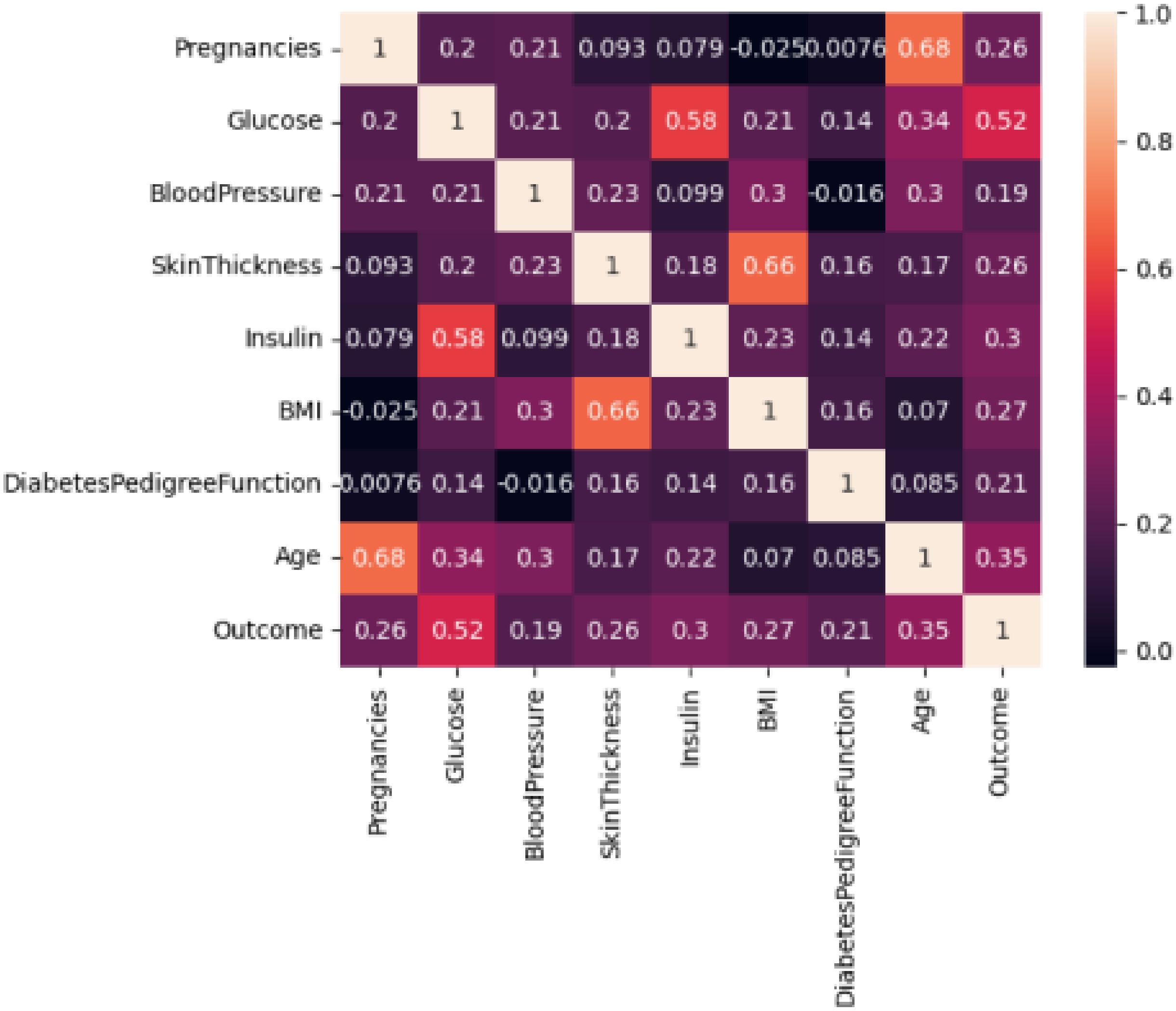
The support vector machine [7] is a very well studied technique. It is a powerful classifier that does not require massive amounts of computational power for normal sized datasets. It is difficult to use on very large datasets due to the matrix math. This dataset is well designed for use with this technique. It is too small for a deep learning systems, but non linear enough to require more than a simple linear classifier.

4. Statistical Inference:

We can download the data and make histrograms of the various parameters. It is clear that there are a number of parameters that are slightly predictive, but are not able to give an predictive value.

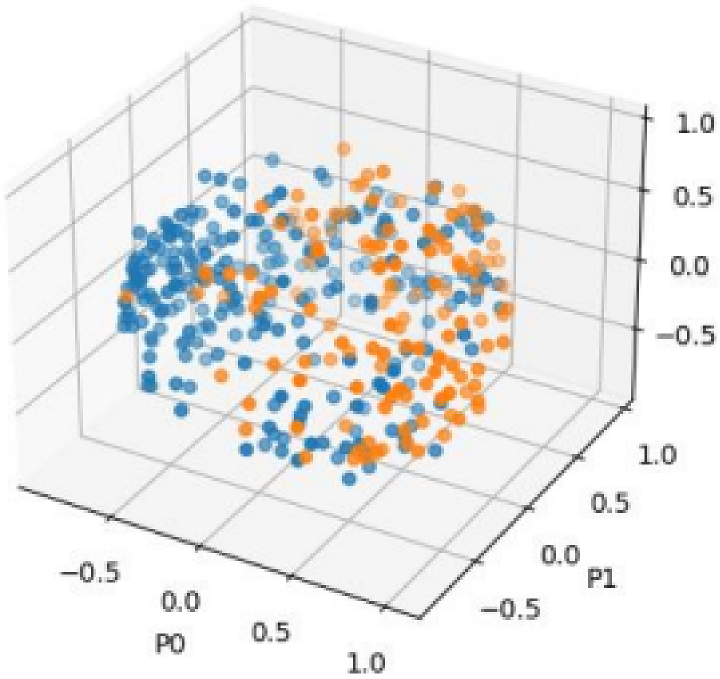


We then do some data cleaning as there is a number of datapoints that have 0 values for a number of individuals, resulting in difficult fits. After the data munging, we can compare the correlation between the features to determine which features may be redundant. We can see that Age and Pregnancies are highly correlated, which is expected. Glucose levels are closely matched with insulin and are the most predictive of all the individual features. The last bit that is very close is BMI and skin thickness, which both are intended to measure the fat content of the body.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.2	0.21	0.093	0.079	-0.025	0.0076	0.68	0.26
Glucose	0.2	1	0.21	0.2	0.58	0.21	0.14	0.34	0.52
BloodPressure	0.21	0.21	1	0.23	0.099	0.3	-0.016	0.3	0.19
SkinThickness	0.093	0.2	0.23	1	0.18	0.66	0.16	0.17	0.26
Insulin	0.079	0.58	0.099	0.18	1	0.23	0.14	0.22	0.3
BMI	-0.025	0.21	0.3	0.66	0.23	1	0.16	0.07	0.27
DiabetesPedigreeFunction	0.0076	0.14	-0.016	0.16	0.14	0.16	1	0.085	0.21
Age	0.68	0.34	0.3	0.17	0.22	0.07	0.085	1	0.35
Outcome	0.26	0.52	0.19	0.26	0.3	0.27	0.21	0.35	1

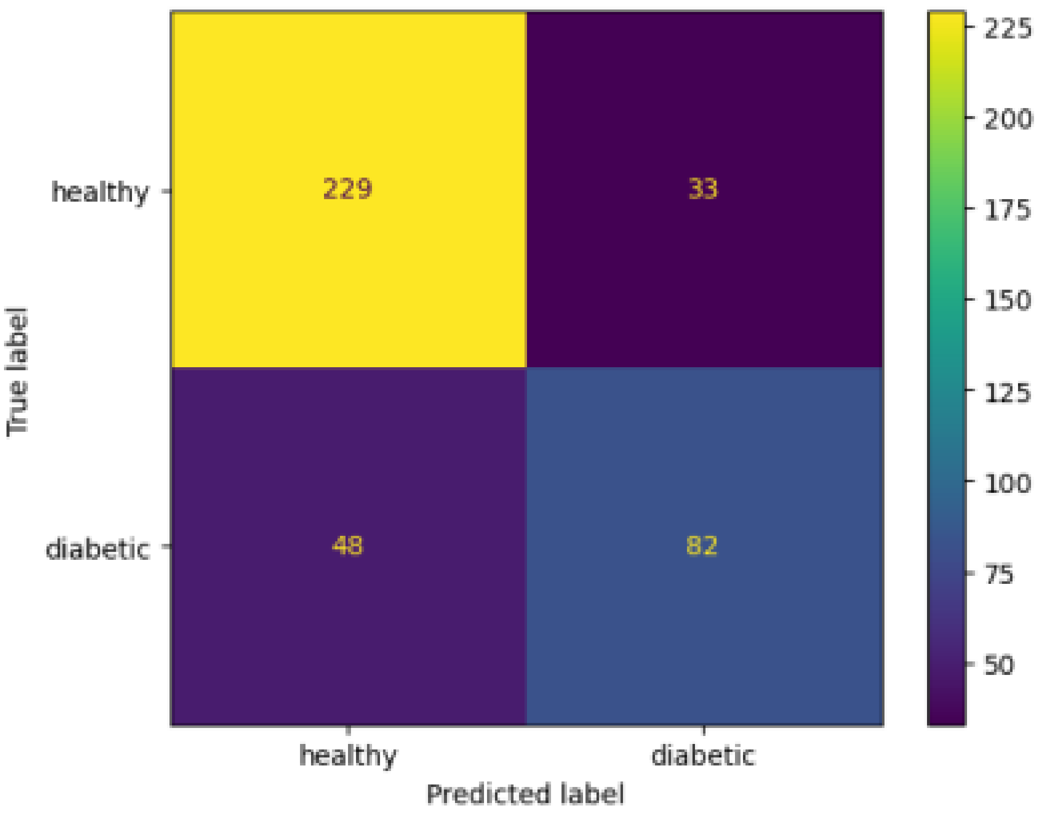
We can combine the columns to create synergy between the columns to combine the various columns. This is done by a technique call principle component analysis. This technique finds the columns that are highly correlated and then rotates the axis to capture the most of both features. This means that the first column has the most variance while the last column has the smallest variance.



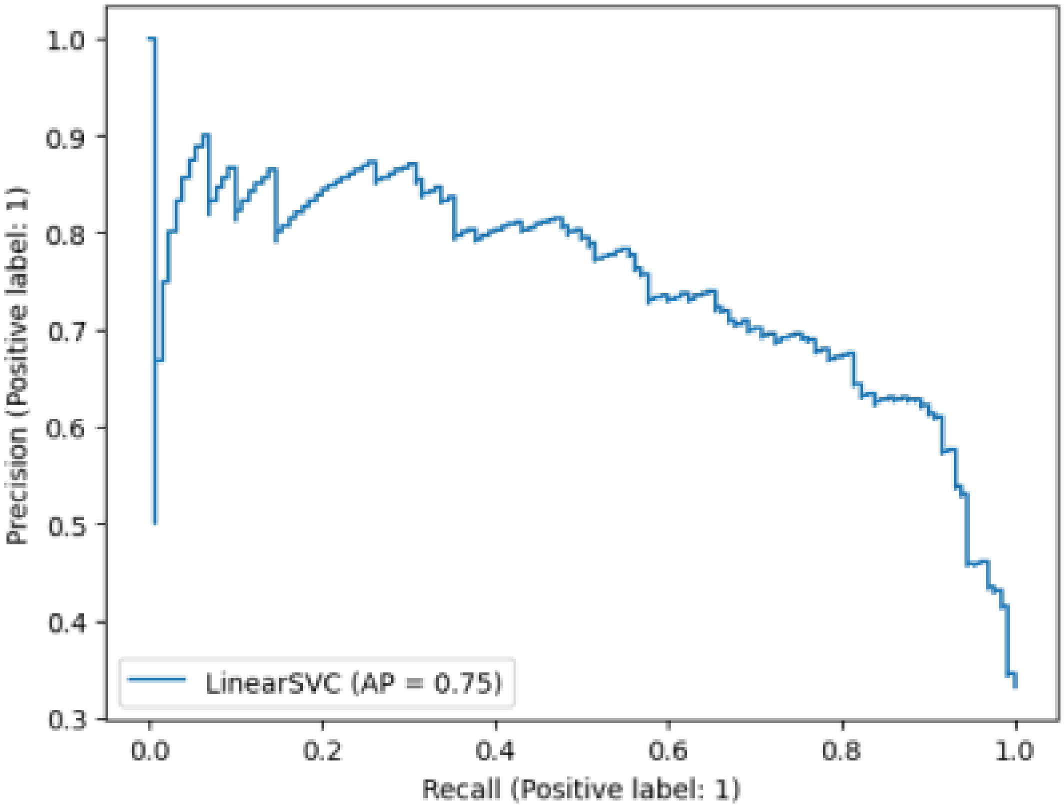
This figure shows the first 3 columns of the PCA transformed data. It can be seen that the data shows a difference between the outcomes. The PCA could be used to perform dimensionality reduction by just cutting off the out columns that have less information.

5. Prediction:

Support vector machines are a machine learning technique that can separate data in classification problems. They come in a variety of flavors which are labeled as



We can determine the accuracy of the code by comparing the correct guesses with the actual data. This gives a score of 80% predictions. This seems to indicate the SVM does a good job of predicting diabetis with this dataset. We can check the precision and recall for the determining the accuracy with a unbalanced dataset.



This indicates that the 80% is fairly robust out through a lot of the different subsets for the data.

6. Ethics

Using artificial intelligence to predict health is a very complicated ethics question. A number of tests have shown that the AI is often able to perform at near the level of a trained physician, but still having a high level of worry about the diagnosis and the lack of human context. A false positive can have lasting affect on the patient and cause undue stress. Many legal and ethical questions must be resolved to allow these products to move into the consumer space as opposed to being controlled by a trained specialist or physician.

7. Conclusion:

The SVM did perform better than the model that was reported in the original paper, by a slight amount. It is clear that not a lot of improvement was allowed, and the benefit was likely from the initial data munging. It would be best to find a better, larger dataset to perform data classifications tests.

8. Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC,NuSVC
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA

url="https://raw.githubusercontent.com/npradaschnor/Pima-Indians-Diabetes
Dataset/master/diabetes.csv"
#Load CSV file using Pandas

pima = pd.read_csv(url)
pima

fig,ax = plt.subplots(5,2, figsize=(10,10))
plt.tight_layout()
ax=ax.flatten()
outcomes = pima['Outcome']
cc=0
for col in pima.columns:
 healthy= pima[col][outcomes==0]
 diabetis= pima[col][outcomes==1]
 ax[cc].hist(healthy, alpha=0.5, label='healthy')
 ax[cc].hist(diabetis, alpha=0.5, label='diabetis')
 ax[cc].legend(loc='upper right')
 ax[cc].set_title(col)
 cc+=1


pima = pima.loc[pima['Insulin'] != 0]

pima = pima.loc[pima['Glucose'] != 0] pima

= pima.loc[pima['BloodPressure'] != 0]

pima = pima.loc[pima['SkinThickness'] != 0]

pima = pima.loc[pima['BMI'] != 0]

pima = pima.loc[pima['Age'] != 0]

pima


fig,ax = plt.subplots(5,2, figsize=(10,10))

plt.tight_layout()

ax=ax.flatten()

outcomes = pima['Outcome']

cc=0

for col in pima.columns:

 healthy= pima[col][outcomes==0]

 diabetis= pima[col][outcomes==1]

 ax[cc].hist(healthy, alpha=0.5, label='healthy')

ax[cc].hist(diabetis, alpha=0.5, label='diabetis')

ax[cc].legend(loc='upper right')

 ax[cc].set_title(col)

 cc+=1


import seaborn as sns
corr_matrix_pearson = pima.corr(method='pearson')
```

```
sns.heatmap(corr_matrix_pearson, annot = True)


outcomes = pima['Outcome']

features = pima.drop(['Outcome'], axis = 1)


scaler = StandardScaler()

features_scaled = scaler.fit_transform(features)

features_scaled = normalize(features_scaled)

features_scaled.shape


pca = PCA(n_components = 8)

X_principal = pca.fit_transform(features_scaled)

X_principal = pd.DataFrame(X_principal)

X_principal.columns = [f'P{i}' for i in range(8)]

X_principal.head()


fig,ax = plt.subplots(5,2, figsize=(10,10))

plt.tight_layout()

ax=ax.flatten()

cc=0

for col in X_principal.columns:

 colData= np.array( X_principal[col])

 healthy = colData[outcomes==0]

 diabetis = colData[outcomes==1]

 ax[cc].hist(healthy, alpha=0.5, label='healthy')
```

```
ax[cc].hist(diabetis, alpha=0.5, label='diabetis')
 ax[cc].legend(loc='upper right')
 ax[cc].set_title(col)
 cc+=1

fig = plt.figure(figsize=(5,5))
plt.tight_layout()
X= np.array( X_principal['P0'])
Y= np.array( X_principal['P1'])
plt.scatter(X[outcomes==0],Y[outcomes==0], label = 'healthy')
plt.scatter(X[outcomes==1],Y[outcomes==1], label = 'diabetis')
plt.legend()
plt.show()

fig = plt.figure()
plt.tight_layout()
ax = fig.add_subplot(projection='3d')

xs= np.array( X_principal['P0'])
ys= np.array( X_principal['P1'])
zs= np.array( X_principal['P2'])

ax.scatter(xs[outcomes==0], ys[outcomes==0], zs[outcomes==0],label = 'healthy')
ax.scatter(xs[outcomes==1], ys[outcomes==1], zs[outcomes==1],label = 'diabetis')

ax.set_xlabel('P0')
ax.set_ylabel('P1')
ax.set_zlabel('P2')
plt.show()


from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
svm=SVC(gamma='auto')
svm.fit(X_principal, outcomes)


predicted = svm.predict(X_principal)
cm = confusion_matrix(outcomes, predicted)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['healthy', 'diabetic'])
disp.plot()


svm = NuSVC(gamma="auto")
svm.fit(X_principal, outcomes)


predicted = svm.predict(X_principal)
cm = confusion_matrix(outcomes, predicted)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['healthy', 'diabetic'])
disp.plot()


from sklearn.metrics import PrecisionRecallDisplay
y_score = svm.decision_function(X_principal)
PrecisionRecallDisplay.from_predictions( outcomes, y_score, name="LinearSVC" )


from sklearn.metrics import accuracy_score
```

accuracy_score(outcomes, predicted)
Bibliography:

1. World Health Organisation (2016) Global Report on Diabetes. https://www.who.int/publications-detail/global-report-on diabetes.Accessed: 24 Apr 2020

2. Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS. Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus. Proc Annu Symp Comput Appl Med Care. 1988 Nov 9:261–5. PMCID: PMC2245318.

3. M. K. Hasan, M. A. Alam, D. Das, E. Hossain and M. Hasan, "Diabetes Prediction Using Ensembling of Different Machine Learning Classifiers," in *IEEE Access*, vol. 8, pp. 76516-76531, 2020, doi: 10.1109/ACCESS.2020.2989857.

4. Larabi-Marie-Sainte S, Aburahmah L, Almohaini R, Saba T. Current techniques for diabetes prediction: review and case study. *Appl Sci.* 2019;9(21):4604. doi: 10.3390/app9214604.

5. https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/code

6. M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. “Support vector machines.” Intelligent Systems and their Applications, IEEE, 13(4), pp. 18–28, 1998.