Don’t Sugar Coat It!

Early-Stage Diabetes Risk Prediction using Artificial Intelligence

Derek Sakasegawa
College of Engineering &
Computer Science
California State University,
Sacramento
Sacramento, CA, USA
dereksakasegawa@csus.edu

Eric Wong
College of Engineering &
Computer Science
California State University,
Sacramento
Sacramento, CA, USA
ericlamwong@csus.edu

Jack Tran
College of Engineering &
Computer Science
California State University,
Sacramento
Sacramento, CA, USA
jacktran@csus.edu

ABSTRACT

Diabetes is a problematic disease that results in high blood sugar levels in the human body. It contributes to many negative health conditions such as blindness, kidney failure, heart attacks, strokes, and lower limb amputation according to the World Health Organization. The problem to solve at hand is to predict if a patient is at risk of diabetes based on their answers to a health questionnaire using artificial intelligence. By being able to predict if one is at risk of diabetes early on, proper treatment and observation can be started as soon as possible. To solve the task, the questionnaire data provided in the U.C. Irvine’s repository was split and trained on various fully connected neural networks and CNN models. In addition, multiple AI algorithms were used to further improve accuracy scores of predictions, including KNN, Random Forest, and SVM. With the various models and algorithms implemented, accuracy scores as high as 99% was obtained for predicting diabetes risk based on the patient’s answers to the questionnaire. Some models had performed poorly, but tuning hyperparameters along with implementing the various algorithms allowed the high accuracy scores to be obtainable.

CCS CONCEPTS

• **Computing Methodologies** → **Machine Learning approaches** → Classification and Regression Tree; Neural Networks;

KEYWORDS

Diabetes, Disease, Prediction, Artificial Intelligence

ACM Reference format:

Derek Sakasegawa, Eric Wong and Jack Tran. 2021. Don’t Sugar Coat It!: Early-Stage Diabetes Risk Prediction using Artificial Intelligence. In *Proceedings of ACM Woodstock conference (WOODSTOCK’18). ACM, New York, NY, USA, 7 pages.* https://doi.org/10.1145/1234567890

1 Introduction

“Don’t Sugar Coat It! Early-Stage Diabetes Risk Prediction using Artificial Intelligence” was completed by Derek Sakasegawa, Eric Wong, and Jack Tran as the final project of CSC 180. For this final project, our team planned on working on COVID-19 related projects as health is a major topic in today’s current condition, but there were too many solutions and not enough different datasets online. With this in mind, our next idea for the project was for it to relate to some kind of early detection of a disease. Upon further research, we found the project of early stage diabetes risk prediction with a dataset published in the UCI Repository in 2020.

The goal of the project that our team wanted to achieve was to predict whether a patient is diabetic or at risk of being diabetic based on the answers he/she provided in a questionnaire. To help us approach this problem, this task will be considered a classification problem. Therefore, various models and algorithms for classification will be utilized to solve this problem.

Contributions:

- Random Forest Algorithm
- K-Nearest Neighbor Algorithm
- Support Vector Machines Algorithm
- Fully-Connected Neural Network Models
- Convolutional Neural Network Models

The organization of the rest of this paper will be as follows: Problem Formulation → System/Algorithmic Design → Experimental Evaluation → Related Work → Conclusion → Work Division → Learning Experience.

2 Problem Formulation

The dataset contains 17 attributes, which are the following: age, sex, polyuria, polysipsia, sudden weight loss, weakness, polyphagia, genital thrush, visual blurring, itching, irritability, delayed healing, partial paresis, muscle stiffness, alopecia, obesity, and class (diabetic or not).

These attributes contain the sign and symptom data of newly diabetic or would-be diabetic patients. All categories besides age, sex and class are either yes or no answers. Age is a numerical value with the range of 20-65, sex is either male or female, and class is positive or negative.

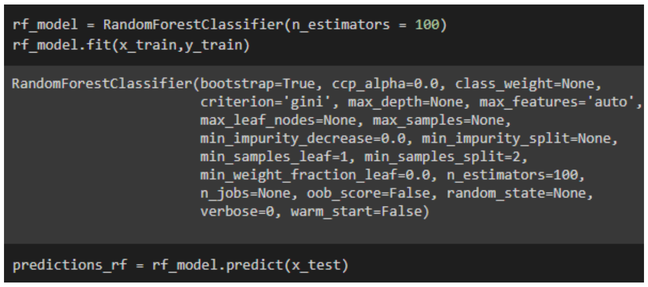
The data described above was used as our inputs, excluding the class attribute, within our project. We used the inputs to train our neural network models to accurately classify a patient as being at risk of developing diabetes or not. Therefore, the outputs of our models would be the predictions of the class attribute of the dataset.

3 System/Algorithm Design

3.1 System Architecture

The design of our project was to include multiple different neural network models as well as algorithms to predict diabetes risk. As a result, a section of the project was dedicated to implementing the algorithms mentioned above, another for creating fully connected neural network models, and finally one for convolutional neural network models (CNN). Rather than having the modules interact with one another, each algorithm or model had its performance evaluated separately. The goal of such a design is to evaluate the performance of each model and to determine which is better for the problem we are trying to address.

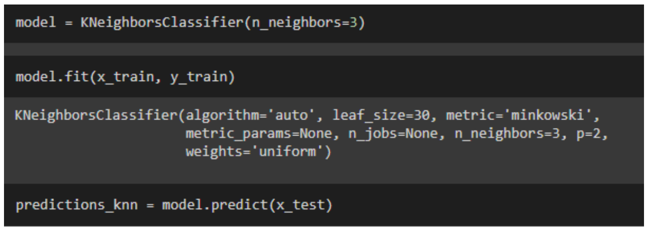
3.2 Algorithms



```
rf_model = RandomForestClassifier(n_estimators = 100)
rf_model.fit(x_train,y_train)

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)

predictions_rf = rf_model.predict(x_test)
```

Figure 1: Random Forest Algorithm Code

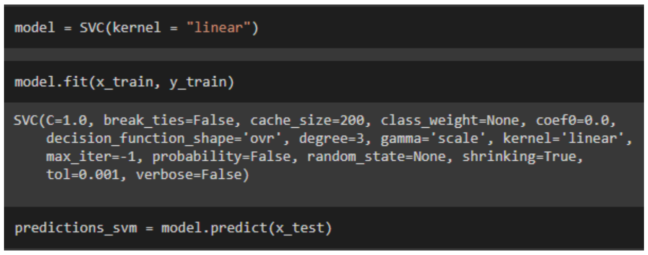
3.2.1 Random Forest Algorithm. The above figure shows the code implementation of the Random Forest Algorithm. [2] The algorithm creates a "forest" of decision trees in which the algorithm uses a combination of the learning models to increase the overall results. In layman's terms, the model learns from the results of the decision trees and obtains an overall best result by looking at which class prediction was the most common from the decision trees. The first line of code initializes our Random Forest model with the parameter of 100 estimators, the next line fits the model with our training data and finally uses the model to predict using testing data.



```
model = KNeighborsClassifier(n_neighbors=3)

model.fit(x_train, y_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')

predictions_knn = model.predict(x_test)
```

Figure 2: KNN Algorithm Code

3.2.2 KNN Algorithm. The above figure shows the code implementation for the KNN Algorithm. The algorithm basically classifies data by selecting the most similar data to it. [2] The algorithm picks a certain data point, selects the closest neighbors (amount specified) to that data, and assigns that picked data to the most common class among its neighbors. The first line of code initializes the model as the KNN algorithm with the parameter of 3 neighbors, fits the model with our training data and finally uses our model to predict.

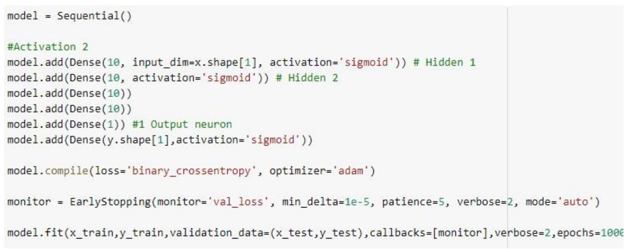


```
model = SVC(kernel = "linear")

model.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=None, shrinking=True,
    tol=0.001, verbose=False)

predictions_svm = model.predict(x_test)
```

Figure 3: SVM Algorithm Code

3.2.3 SVM Algorithm. The above figure shows the code implementation for the SVM Algorithm. [2] The SVM algorithm creates a decision boundary, a hyperplane that separates two classes of data, with the largest possible distance to data points. SVMs try to pick the ones that are closest to the decision boundary as key components for the classifier. The first line of code initializes the model as the classification SVM algorithm with the parameter of the kernel being linear, fits the model with the training data and finally uses our model to predict with the testing data.

3.3 Fully Connected Neural Network Models

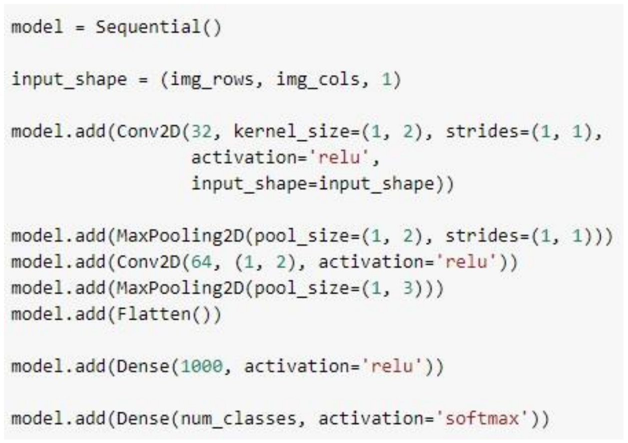


```
model = Sequential()

#Activation 2
model.add(Dense(10, input_dim=x.shape[1], activation='sigmoid')) # Hidden 1
model.add(Dense(10, activation='sigmoid')) # Hidden 2
model.add(Dense(10))
model.add(Dense(10))
model.add(Dense(1)) #1 Output neuron
model.add(Dense(y.shape[1],activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam')

monitor = EarlyStopping(monitor='val_loss', min_delta=1e-5, patience=5, verbose=2, mode='auto')

model.fit(x_train,y_train,validation_data=(x_test,y_test),callbacks=[monitor],verbose=2,epochs=100
```

Figure 4: Example Code for Fully Connected Neural Network Model

3.3.1 Design. No algorithm was implemented for the fully connected neural network models. Instead, they were built from scratch using various activations and optimizers. In addition, dense layers with varying neuron counts were added for the sake of evaluating performance. Figure 4 showcases a snippet of code for building one of the six fully connected neural network models that we built. For this specific model, it used the sigmoid activation with the adam optimizer. Early stopping was also implemented to prevent overtraining of this model and all other ones. Then it was fitted to our model and then trained for testing. All fully connected neural network models had their hyperparameters tuned, including the one shown in Figure 4.

3.4 Convolutional Neural Network Models



```
model = Sequential()

input_shape = (img_rows, img_cols, 1)

model.add(Conv2D(32, kernel_size=(1, 2), strides=(1, 1),
                 activation='relu',
                 input_shape=input_shape))

model.add(MaxPooling2D(pool_size=(1, 2), strides=(1, 1)))
model.add(Conv2D(64, (1, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(1, 3)))
model.add(Flatten())

model.add(Dense(1000, activation='relu'))

model.add(Dense(num_classes, activation='softmax'))
```

Figure 5: Example Code for Convolutional Neural Network Model

3.4.1 Design. No algorithm was implemented for the convolutional neural network models. In addition, the CNN models did not use any transfer learning before training our dataset. Instead, they were built and trained from scratch. Figure 5 shows the code we implemented for one of our CNN models. As seen, multiple Conv2D and MaxPooling2D layers were added. The kernel size and numbers were adjusted for the sake of testing performance. Dense layers were also added with activations during the building of the model. Unlike the fully connected neural network models, the main focus of design for the CNN models was changing their kernel sizes and numbers to evaluate performance. The activations were not as significant as they were in the other models.

4 Experimental Evaluation

4.1 Methodology

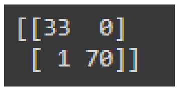
The dataset containing patients’ answers to a health questionnaire provided by the UCI Repository was used. As mentioned in the Problem Formulation section above, there were 17 attributes tied to each patient. As a reminder, the attributes were age, sex, polyuria, polysipsia, sudden weight loss, weakness, polyphagia, genital thrush, visual blurring, itching, irritability, delayed healing, partial paresis, muscle stiffness, alopecia, obesity, and class (diabetic or not). To split the data for training and testing, we first classified the inputs (x) and the outputs (y). All attributes except for class were set as the inputs and then class was set as the output. Next, we divided the inputs and outputs into an 80/20 split for training and testing (80% training size and 20% testing size).

The experimental setting was a combination of local machines and Google Colab. To ensure no memory issues, the project was tested on Jupyter Notebook on any one of our machines since each of us has high RAM available. For collaborative work, the project was also run on Google Colab to help with performance issues if there were any. To use the data on our personal computers, the data was located locally in the same location as the notebook file. However, for Colab, the data had to be constantly uploaded each time if a session ended.

Regarding comparison of the performance of our models and algorithms, the metrics used included confusion matrices and classification reports with precision, recall, and F1 scores. The metrics used helped us examine the accuracy scores based on each model or algorithm’s performance which allowed us to determine which is better in accurately predicting diabetes risk.

Methods implemented for training our neural network models include early stopping and hyperparameter tuning. The main point of comparison was in regards to the accuracy scores based on the performance of each model whose parameters were tuned separately. On the side of fully connected neural network models, different activations (relu, sigmoid, and tanh) and optimizers (adam and sgd) were used. In addition, various dense layers with different neuron counts were added upon each of those models to see if its performance can be improved upon. For our CNN models, the kernel size and number were tuned to see how performance would be affected. In addition, layer count and their respective activations were tuned as well.

4.2 Results



```
[[33  0]
 [ 1 70]]
```

Figure 6: Numeric Values of CM for Random Forest

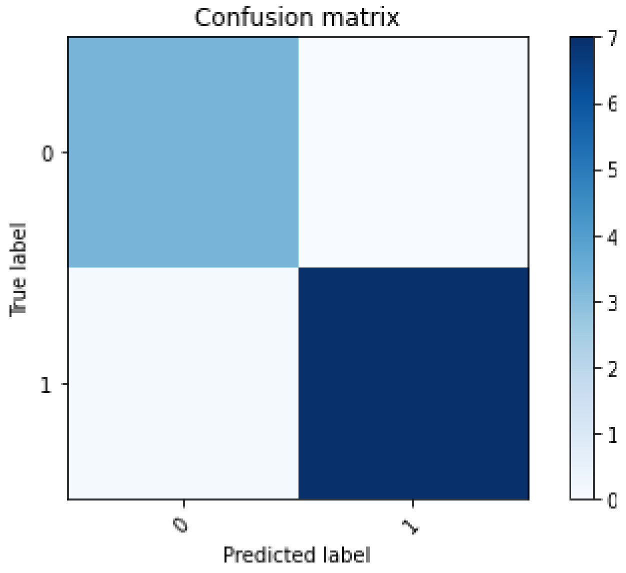
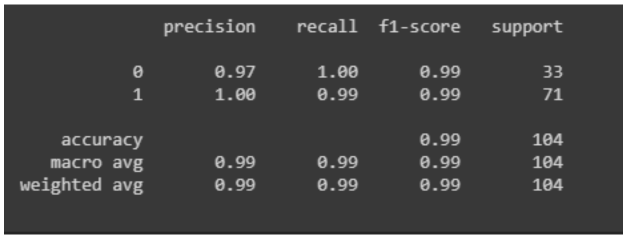


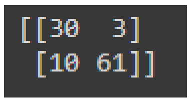
Figure 7: CM for Random Forest



```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        33
           1       1.00      0.99      0.99        71

    accuracy                           0.99       104
   macro avg       0.99      0.99      0.99       104
weighted avg       0.99      0.99      0.99       104
```

Figure 8: Classification Report for Random Forest

4.2.1 Random Forest Algorithm. Figures 6-8 are the results of the predictions made by the Random Forest Algorithm model. It achieved an overall accuracy of 0.9903846153846154 and it predicted all but 1 person incorrectly. The model incorrectly predicted 1 diabetic person as a non-diabetic as shown by the confusion matrices above. Overall, the model was able to correctly recall all non-diabetics and had a 99 recall for predicting diabetic people. Overall, the Random Forest model was able to achieve a very high overall accuracy, only misclassified all but 1 person.



```
[[30  3]
 [10 61]]
```

Figure 9: Numeric Values of CM for KNN

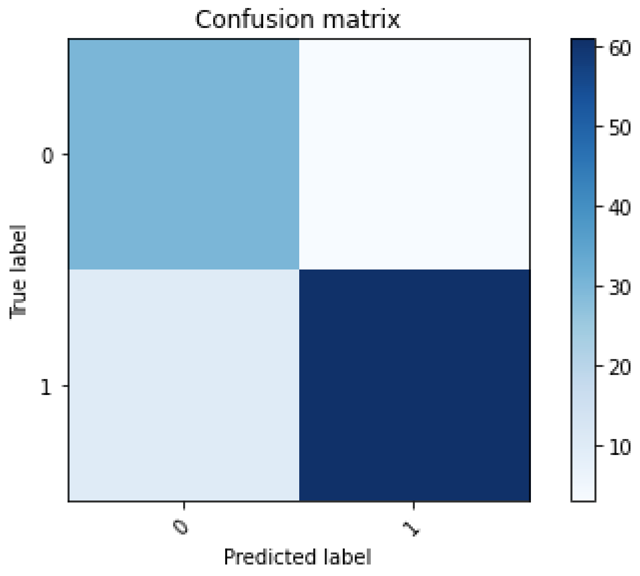
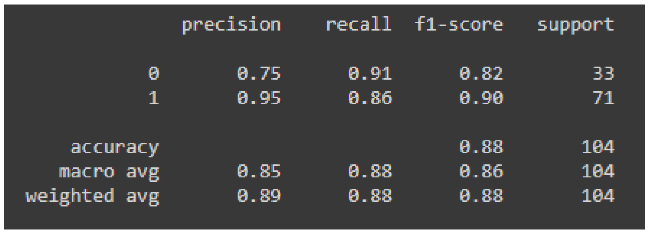


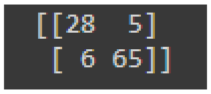
Figure 10: CM for KNN



```
              precision    recall  f1-score   support

           0       0.75      0.91      0.82        33
           1       0.95      0.86      0.90        71

    accuracy                           0.88       104
   macro avg       0.85      0.88      0.86       104
weighted avg       0.89      0.88      0.88       104
```

Figure 11: Classification Report for KNN

4.2.2 KNN Algorithm. Figures 9-11 are the results of the predictions made by the KNN Algorithm model. The final accuracy of the KNN model achieved was 0.875. This model, compared to the Random Forest model, was overall less accurate and was the worst out of the three algorithms that we implemented. Overall, the KNN accurately predicted 91 patients out of 104 of the test data. It incorrectly predicted 10 diabetic patients as non-diabetics and predicted 3 non-diabetics as diabetics. The recall score of diabetic patients was 86, while the recall score of non-diabetic patients was 91.



```
[[28  5]
 [ 6 65]]
```

Figure 12: Numeric Values of CM for SVM

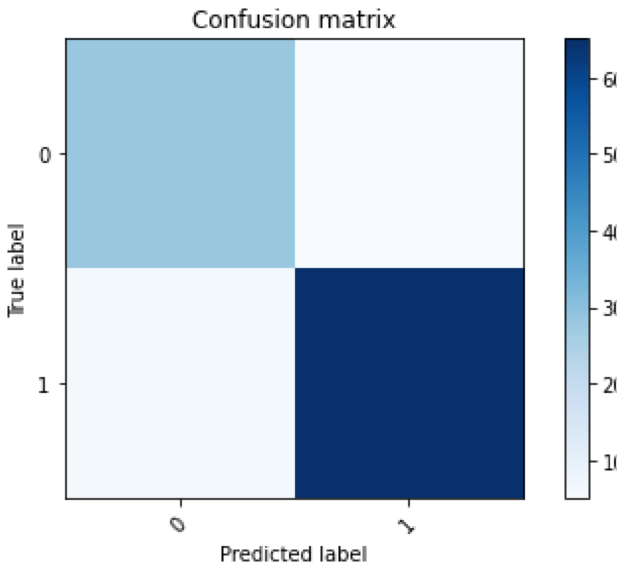
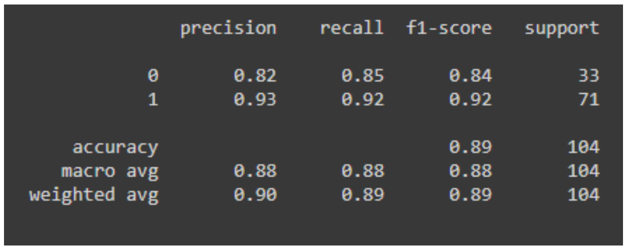


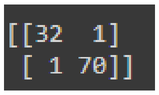
Figure 13: CM for SVM



```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        33
           1       0.93      0.92      0.92        71

    accuracy                           0.89       104
   macro avg       0.88      0.88      0.88       104
weighted avg       0.90      0.89      0.89       104
```

Figure 14: Classification Report for SVM

4.2.3 SVM Algorithm. Figures 12-14 are the results of the predictions made by the SVM Algorithm model. The overall accuracy of SVM was 0.8942307692307693. Out of the 104 patients used in our testing data, the SVM algorithm was able to correctly predict 94 patients as being diabetic or non-diabetic. The recall score of diabetic patients was 92, while the recall score of non-diabetic was 85. The SVM algorithm incorrectly predicted 5 non-diabetics to be diabetics and 6 diabetics as non-diabetics. Overall, it achieved a higher accuracy score than the KNN Algorithm, but failed to pass the Random Forest Algorithm in overall accuracy.



```
[[32  1]
 [ 1 70]]
```

Figure 15: Numeric Values of CM for Fully Connected Neural Network Model

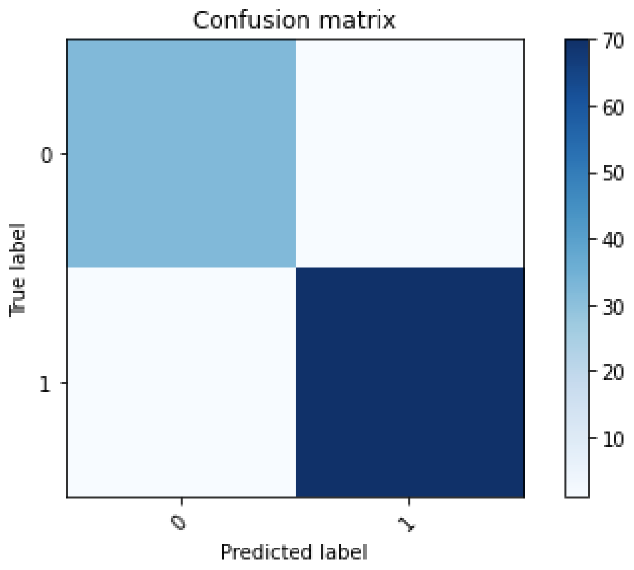
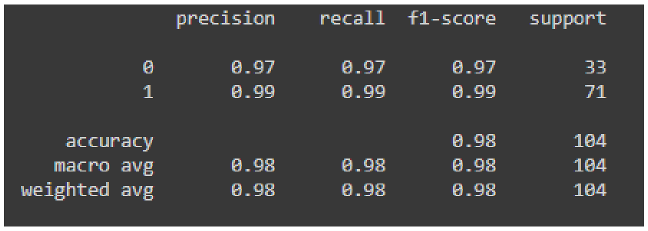


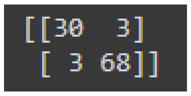
Figure 16: CM for Best Fully Connected Neural Network Model



```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        33
           1       0.99      0.99      0.99        71

    accuracy                           0.98       104
   macro avg       0.98      0.98      0.98       104
weighted avg       0.98      0.98      0.98       104
```

Figure 17: Classification Report for Best Fully Connected Neural Network Model

4.2.4 Fully Connected Neural Network Models. Figures 15-17 are the metrics we implemented for the best fully connected neural network model of the six that we created. Figure 15 shows the numeric values of what the model predicted. So it predicted 32 patients as being non-diabetic and 70 as being diabetic correctly. However, it incorrectly predicted one patient as being diabetic when they were in fact non-diabetic. The same can also be said for the incorrect prediction for a patient being non-diabetic. That person was indeed diabetic or at risk of being diabetic. In other words, our best fully connected neural network model predicted 102 patients' condition correctly out of 104 in total. Figure 17 is the metrics report on the performance of our best model. The accuracy score was 98% which was very high compared to the rest of the other models. When tuning the hyperparameters of these models, we found the models' performances to be very volatile. The accuracy scores would range anywhere from 30% to 70% consistently, with a few exceptions reaching higher scores like our best model. Due to the inconsistent performances of the fully connected neural network models, our team determined that they should not be considered the most optimal method of predicting diabetes risk based on the dataset used.



```
[[30  3]
 [ 3 68]]
```

Figure 18: Numeric Values for CM of Best CNN Model

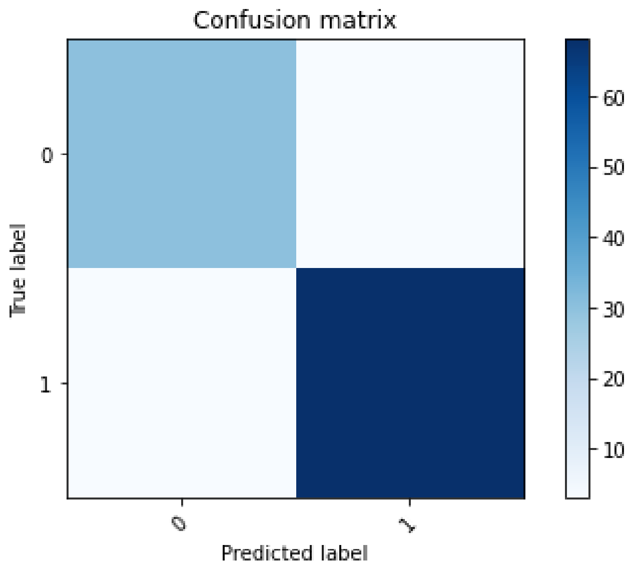
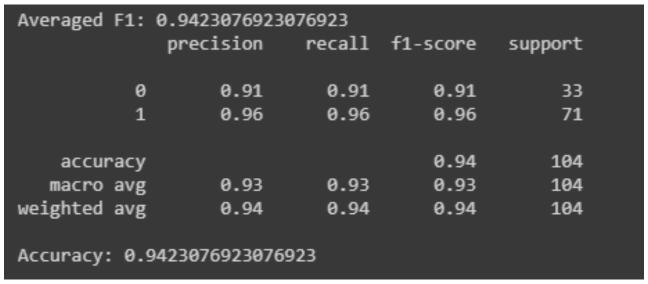


Figure 19: CM for CNN



```
Averaged F1: 0.9423076923076923
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        33
           1       0.96      0.96      0.96        71

    accuracy                           0.94       104
   macro avg       0.93      0.93      0.93       104
weighted avg       0.94      0.94      0.94       104

Accuracy: 0.9423076923076923
```

Figure 20: Classification Report for CNN

4.2.5 CNN Models. Figured 18-20 are the metrics implemented for our best CNN model of the five that we created. Figure 18 shows the numeric values predicted correctly and incorrectly for each class (non-diabetic or diabetic) of all the patients. Our CNN model was able to correctly predict 30 patients as being non-diabetic and 68 patients as being diabetic. However, it incorrectly predicted 3 patients as being non-diabetic and 3 as being diabetic. All in all, it predicted 98 correctly out of the 104 patients in total. Figure 20 is the metrics report on the performance of our best CNN model with an accuracy score of 94%. When tuning the kernel sizes and numbers of our five models, we found that the accuracy score varied between 80% to above 90%. Unlike the fully connected neural network models, the CNN models were more consistent in showing better performances all around. As a result, our team believed that CNN models should be the optimal solution in predicting diabetes risk based on the dataset used.

5 Related Work

In [3], their problem revolves around predicting early stage diabetes by using significant attributes, and characterizing the relationship of the differing attributes. Principal component analysis was used to select their significant values. The methods they used to predict included: Artificial Neural Network (ANN), Random Forest (RF) and K-means clustering techniques. The attributes that this academic research paper chose are the following: pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, age and outcome. Our methods, when compared to the methods used in the academic paper, differed as we included multiple different algorithms. For example, within our project we included the following methods for our prediction models: Random Forest, SVM, KNN, Fully-Connected Neural Networks, and Convolutional Neural Networks. Our method in solving this problem is better compared to the academic paper as we included more than just machine learning algorithms. We constructed and built the Fully-Connected Neural Networks as well as the Convolutional Neural Networks. Also, our methods achieved overall higher accuracies. The overall high accuracy that the research paper achieved was 75.7%, but this could be due to a number of different variables.

In [4], their problem revolves around using big data analytics to create a diabetes prediction model for classification. Their method involves using various machine learning algorithms on a dataset of 800 records with 10 attributes including glucose level, blood pressure, age, BMI, outcome, and much more. Their architecture incorporates five modules: (1) dataset collection, (2) data pre-processing, (3) clustering, (4) build model, and (5) evaluation. Our team's problem at hand is somewhat similar in that we both want to use artificial intelligence to predict diabetes classification. However, the dataset that we use in comparison to what is used inside [4] is different; ours is obtained from the UCI Repository and is much smaller. Furthermore, our method is different as we utilize neural network models and put more emphasis on those in comparison to the machine learning algorithms that we implemented. We used Random Forest, SVM, and KNN as did [4], but the accuracy scores we obtained differ as well. Since [4] did not build any fully connected neural network models or CNN models, our method is better in that we did more than just implement machine learning algorithms. They incorporated a larger number of algorithms, but their accuracy scores for the ones that we also implemented (Random Forest, KNN, and SVM) are overall lower than ours. [4] achieved an accuracy score of 91% for Random Forest, 90% for KNN, and 60% for SVM/SVC. On the other hand, we obtained 99% for Random Forest, 87.5% for KNN, and 89% for SVM.

6 Conclusion

For the three algorithms implemented in our final project, the Random Forest achieved the highest overall accuracy of 0.9903846153846154. The SVM Algorithm, the second highest accuracy out of the three models, achieved an accuracy of 0.8942307692307693. Finally, the KNN model, the worst of the three, achieved an overall accuracy of 0.875. While implementing these three algorithms, our team changed the parameters mentioned in each of their respective sections earlier in the report. For the Random Forest model, we tuned the amount of estimators to be 100. If we decreased this value, our accuracy would decrease significantly. If we increased this value, the overall accuracy didn't change or had very little change to it. For the KNN algorithm, we found the best amount of neighbors to achieve the highest overall accuracy was 3. Increasing the values would decrease our overall accuracy. Finally, the SVM kernel was the parameter we changed to achieve a higher overall accuracy. The default kernel of SVC, the classification algorithm of SVM, is "rbf" or "Radial Basis Function". When implementing the "rbf" kernel, our overall accuracy was about .35. However, once we changed the kernel to be "linear", our overall accuracy increased.

Between the two neural network model types implemented, the fully connected neural network model had the highest accuracy score with 98% compared to the CNN model's 94%. However, the performance of the fully connected neural network models we created were very inconsistent. Tuning the hyperparameters would significantly alter the performance for better or worse with accuracy scores dropping as low as 30% and no higher than the 98% we managed to achieve. We found that the fully connected neural network models would consistently fall in the 30%-60% range in terms of accuracy. On the other hand, our CNN models were much more consistent in their performance with accuracy scores hovering at above 80% on each trial run. This leads us to believe that using CNN models would be the better predictor of diabetes risk in patients compared to fully connected neural network models. Although our best CNN model's accuracy score was lower than the fully-connected neural network model's, its consistency in performing with an accuracy score of at least 80% led to us deciding that CNN is the better option.

All in all, the task at hand was implemented and completed successfully with the various algorithms and models we wanted to use. Not only were we able to use artificial intelligence to predict diabetes risk in patients, but we also managed to evaluate the performance between fully connected neural network models and CNN models.

7 Work Division

For code writing and the report, it was all done collaboratively without any true division. Dates and times were set up so we could meet and code in real time and discuss any issues on a third party voice chat known as Discord. For coding, one team member would share their screen and write code while the other two would research for any examples to reference from or solutions to problems that arise. Every so often, we would rotate tasks. A new person would code next while the other two focused on research. This ensured that each team member contributed equal amounts of work. For the report, all members worked on Google Docs to fill out all required sections. Every section was looked at by all members and written in a manner that we all agreed upon.

8 Learning Experience

This final project shows our accumulated knowledge of what we have learned about neural networks and how to predict with them. We as a team learned to translate our previous projects techniques and processes to solve a new problem of diabetes prediction. Our team also gained new knowledge of other algorithms used for classification problems such as the Random Forest Algorithm, the KNN algorithm, and the SVM algorithm. Finally, the last learning experience we gained from this project was on creating a pseudo academic paper for publication. Overall, the project required and tested us into applying our learned knowledge of neural networks on a new topic of diabetes prediction, while also helping us learn new algorithms in the process.

ACKNOWLEDGMENTS

This project was completable thanks to course material provided by Dr. Haiquan Chen's CSC 180 Intelligent Systems course and Dr. Meiliu Lu's CSC 177 Data Mining course. In addition, the dataset provided by the UCI Repository allowed the models and algorithms to be properly trained and tested.

REFERENCES

[1] Haiquan (Victor) Chen, "CSC 180 Intelligent Systems" (CSC 180 Intelligent Systems, California State University, Sacramento, Spring 2021).

[2] Meiliu Lu, "CSC 177 Data Analytics and Mining" (CSC 177 Data Analytics and Mining, California State University, Sacramento, Spring 2021).

[3] Talha Mahboob Alam, Muhammad Atif Iqbal, Yasir Ali, Abdul Wahab, Safdar Ijaz, Talha Imtiaz Baig, Ayaz Hussain, Muhammad Awais Malik, Muhammad Mehdi Raza, Salman Ibrar, Zunish Abbas, 2019 "A model for early prediction of diabetes", Informatics in Medicine Unlocked. Elsevier Ltd. DOI: https://doi.org/10.1016/j.imu.2019.100204

[4] Aishwarya Mujumdar, V Vaidehi Dr., 2019 "Diabetes Prediction using Machine Learning Algorithms". *Procedia Computer Science.* Elsevier Ltd. DOI: https://doi.org/10.1016/j.procs.2020.01.047.