ML Application In Healthy Living Partner Mobile App

Machine Learning Engineering Project

Francis Odo

STATUS: First DRAFT

Overview

Costs of healthcare continue to rise at a significantly high rate in the United States with no limit in sight. More troubling for patients and the healthcare industry is the cost of diagnosing chronic diseases. Healthcare providers, patients and insurance service providers are faced with this very challenging problem.

Healthcare Service Providers have expressed interests in finding solutions that are based on preventative approaches to help patients through increased and consistent physical activities that will achieve healthy habits and establish a long-term healthy lifestyle.

The payback can be described two folds, improved patient health condition and reduced cost of diagnostics and treatment. Potentially, it could be a win-win situation for patients, healthcare service providers, pharmaceutical industries and the insurance carriers.

Overall, Healthcare Service Providers want to decrease spending on chronic diseases such as type 2 diabetes and related conditions.

The approach adopted in this document is to propose a product solution and articulate the application of Machine Learning techniques using a classification model within a supervised learning architecture. The scope is limited Machine Learning as a decision support component within a product solution architecture.

Problem Statement

Chronic disease such as Type-2 Diabetes is affecting quality of lives significantly across age categories. Research shows that this disease often affects major arteries in humans. It causes damage to large and small blood vessels. Most often, problems extend to deterioration of the kidney and vision performance. Due to the nature of the disease and how widely spread in the society, the cost of diagnosis and treatment is increasingly high. Healthcare and pharmaceutical industries are faced with uncontrollable costs, projecting 26% increase over a five year period.

The most viable and cost effective preventative approach to reducing and possibly eliminating this staggering cost problem is through controlled healthy habits.

This can be achieved with a Machine Learning binary classifier algorithm such as the Logistic Regression model and Neural Network (TensorFlow/Keras) model. The algorithm works as the decision support core component of the application.

There are a number of research and articles in this area of ML/AI in Healthcare particularly for identifying chronic illness.

Healthcare providers can save significantly on costs to diagnose and treat less severe cases.

Metrics

Minimum Viable Product(MVP) that incorporates machine Learning algorithms for intelligent decision support systems. Decisions are transformed into recommendations for the user as a means of achieving a healthy lifestyle.

Evaluation of Machine Learning performance will depend on:

1. Confusion Matrix
2. Precision
3. F1-Score
4. Recall
5. Accuracy of prediction 70% or higher

Analysis

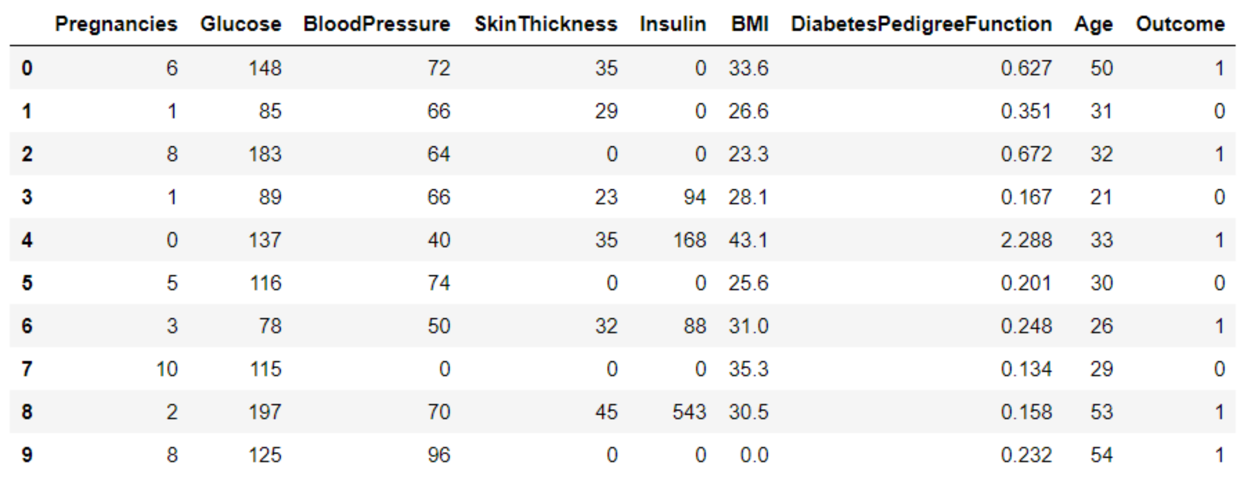
Ideally, users of the app will consistently generate data over a period of time if committed to using the app and following the guidelines offered. The data is then used to train a model, which can be deployed and used to make predictions with respect to health status classification and trends in health conditions.

Since this proposal doesn't have its own generated data, I am using sample data obtained from KAGGLE for this application. The data is CSV formatted (diabetes.csv). The difference will be that the data generated by the user reflects personal activity and performance within the app itself, compared to a general sample with a similar set of features.

Data Exploration

The nature and characteristics of the dataset consists of 768 data points and 9 features. We are only interested in features that are contributing factors to the development of Diabetes disease that the Machine Learning Algorithms can train on. Among them are:

- Glucose - Measures the type of sugar you get from the food you eat
- BloodPressure - The pressure of the blood within the Arteries
- Insulin - Scores the regulation of glucose level
- DiabetesPedigreeFunction - Measures likelihood of diabetes based on family history
- BMI - Blood Mass Index
- Outcome - Indicates Diabetic with (1) and Non-Diabetic with (0)



	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Fig-1 DataFrame - Diabetic.csv

The source of data is the user and user activities. As the user participates, data is created and generated daily across the app and warehoused. The goal is to operate with each of the features set at a reasonable level that will enhance healthy living.

This dataset serves as input to the Machine Learning component of the solution. It is important to note that this data is taken from pregnant women only. The dataset does not represent the general population. In the absence of the real user generated data, It is meant to demonstrate the application concept in the product solution.

Data Visualization

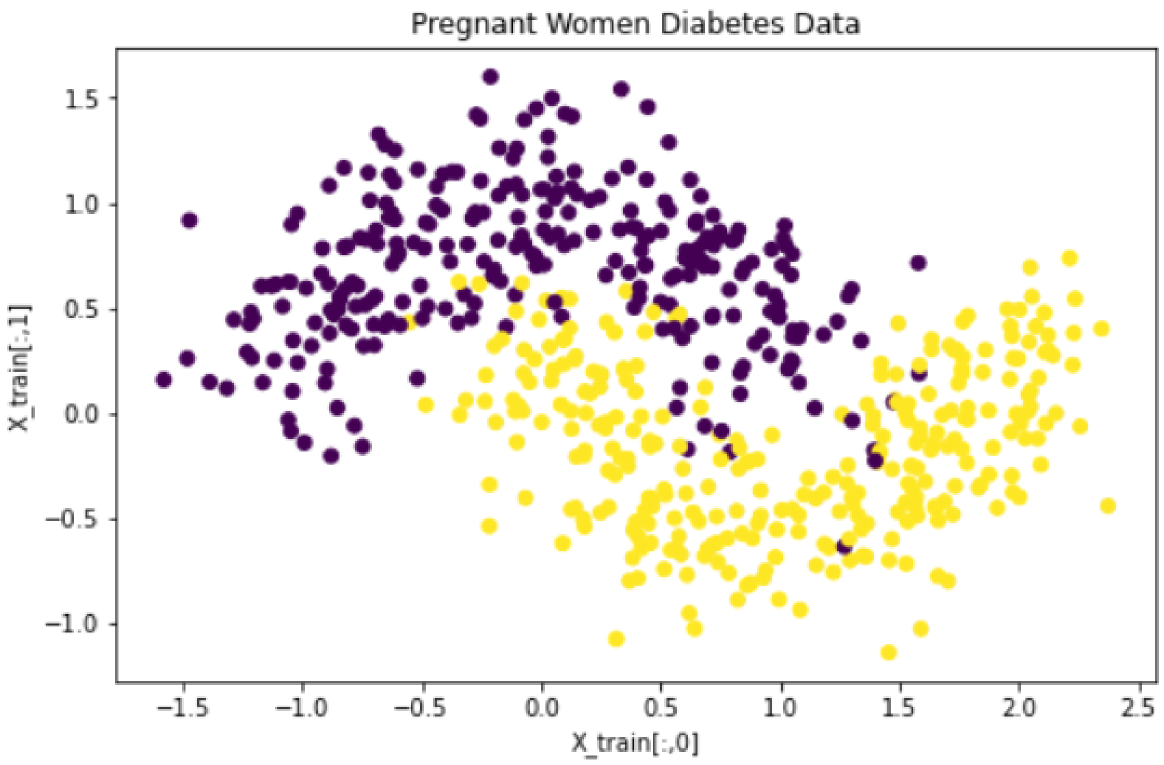


Fig-2 Showing data point distribution classes (diabetic.csv)

Using Matplotlib plotting capabilities, the figure shows the distribution of possible Y_train classes for the dataset. Purple color indicates (0) for non-diabetic while Yellow color (1) represents diabetic.

Methodology

Implementation

The focus of this project is to show how Machine Learning techniques are applied in the product solution as a decision support component.

This product solution is a web-based mobile application which requires hosting. The software development of the core features is not included in this proposal. The code sample provided along with this proposal is limited to the Machine Learning application aspect only. However, the concept and structure of the product solution key features are also articulated.

Environment & Development Tools

A. Scikit Learn
B. Python/Pandas
C. TensorFlow
D. MatplotLib
E. Jupyter Notebook
F. Figma - Design and user workflow
G. Visual Studio Code (For HTML/Python code development)
H. Flask - Web Server/Framework(For complete implementation and deployment)
I. Python 3.0

Code Outline

1. Read in the csv formatted dataset as dataframe
2. Explore and understand the characteristics/nature of the dataset
3. Perform data cleaning as needed
4. Split training/test datasets
5. Preprocess numerical data using StandardScaler
6. Define the Logistic Regression model
7. Train the model
8. Evaluate the model(determine the accuracy)
9. Optimize, if and where necessary

The code can found here:
https://github.com/Francodo/Machine-Learning-Engineering/blob/main/healthy-living-partner.ipynb

Data Preprocessing

The data is fairly clean for the most part. However, some cleaning and rearrangement are required as part of preprocessing as listed:

1. Read in the csv data (diabetes.csv)
2. Inspect data features (filling nulls, drop NaNs, drop Skin Thickness and the Target column).
3. Normalize if we need to change values to numeric(only if necessary)
4. Get the data points and standardize with standardscaler() i.e remove the mean and scale according to standard deviation
5. Obtain dataset class distribution and balance the dataset
6. Create visualization to see the distribution's graphical representation

7. Split into training and testing

The Machine Learning process involves these steps:

1. Define the ML model
2. Train the model
3. Evaluate the model

Machine Learning Algorithms

1. Logistic Regression

   The Logistic Regression binary classifier predicts outcomes where there are only two possibilities as in (yes) or (no). It is a supervisedMachine. Dataset is divided into FEATURES(X) and TARGET(Y). Features are the inputs to making predictions. Target is the outcome.
   In theory, the logistic function g(z) otherwise known as the sigmoid function is described by:

```
g(z) = 1/(1 + exp(-z)

As the value of z increases, g(z) goes to 1
As the value of z decreases, g(z) goes to 0
```

   Graphically represented as:

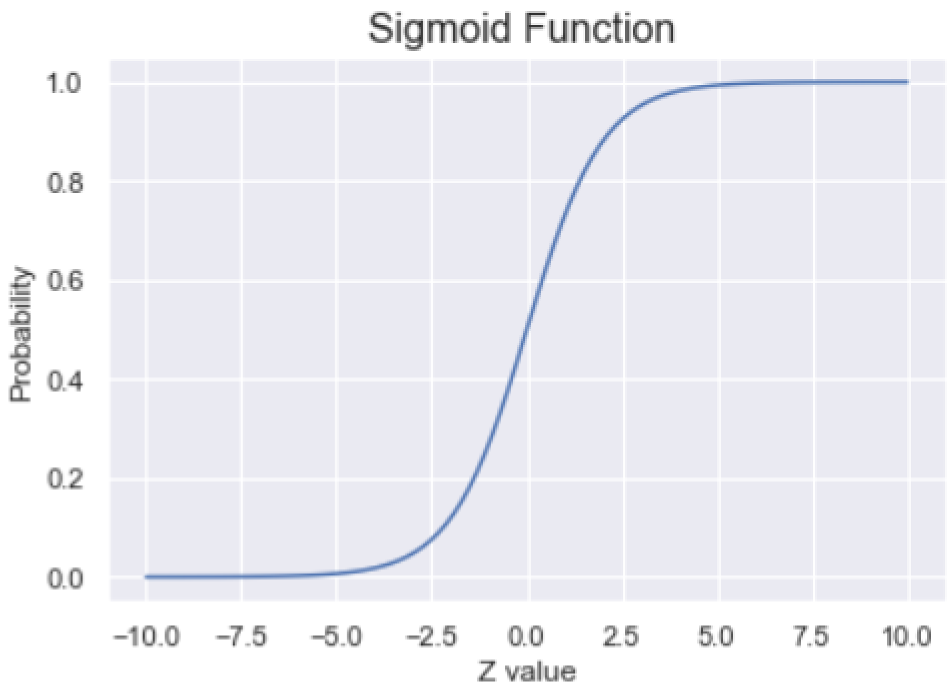
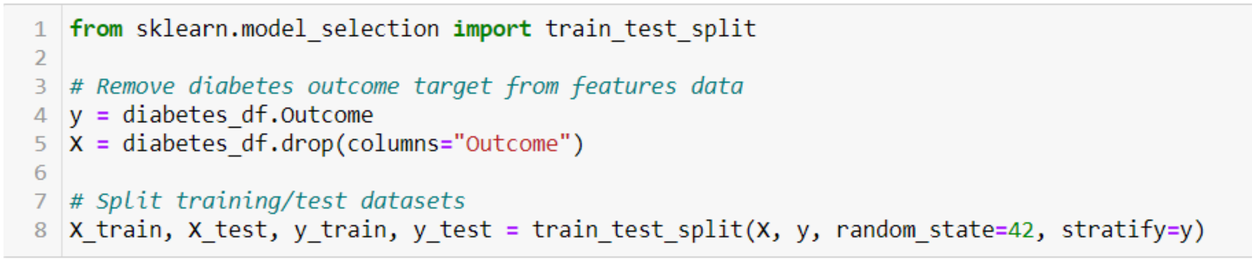


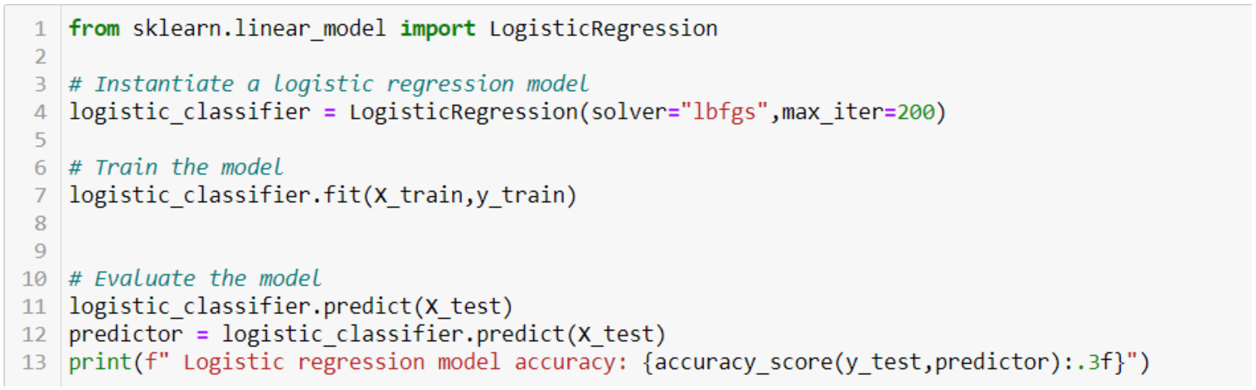
Fig-3 The Logistic Regression - Sigmoid Function

Split the dataset into training and test set (after the necessary preprocessing)



```python
from sklearn.model_selection import train_test_split

# Remove diabetes outcome target from features data
y = diabetes_df.Outcome
X = diabetes_df.drop(columns="Outcome")

# Split training/test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, stratify=y)
```

Create the model, Train the model and Evaluate the model



```python
from sklearn.linear_model import LogisticRegression

# Instantiate a logistic regression model
logistic_classifier = LogisticRegression(solver="lbfgs",max_iter=200)

# Train the model
logistic_classifier.fit(X_train,y_train)


# Evaluate the model
logistic_classifier.predict(X_test)
predictor = logistic_classifier.predict(X_test)
print(f" Logistic regression model accuracy: {accuracy_score(y_test,predictor):.3f}")
```

2. Neural Network

Neural Network or Artificial Neural Network are a series of algorithms designed to operate the same way the human brain processes and analyzes information. It could also be described as an interconnected group of nodes of a layered network of neurons that are artificial in nature.

The major components are:

- Input Layer - input values transformed by weight coefficients
- Hidden Layer - single or multiple neurons
- Output Layer - that reports classification

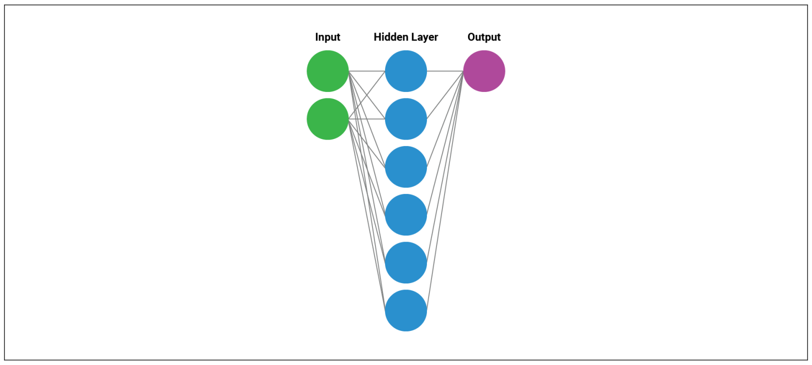


Fig-4 Neural Network

Normalize or standardize our numerical variables

```python
# Preprocess numerical data for neural network. Normalize or standardize
# our numerical variables to ensure that our neural network does
# not focus on outliers and can apply proper weights to each input.

from sklearn.preprocessing import StandardScaler

# Create a StandardScaler instance
scaler = StandardScaler()

# Fit the StandardScaler
X_scaler = scaler.fit(X_train)

# Scale the data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

Create the Neural Network model, compile, train and evaluate

```python
# Define the basic neural network model

nn_model = tf.keras.models.Sequential()
nn_model.add(tf.keras.layers.Dense(units=16, activation="relu", input_dim=8))
nn_model.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Compile the Sequential model together and customize metrics
nn_model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])

# Train the model
fit_model = nn_model.fit(X_train_scaled, y_train, epochs=100)

# Evaluate the model using the test data
model_loss, model_accuracy = nn_model.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

3. Ensemble and AdaBoost Classifier

   The ensemble AdaBoost classifier is a combination of multiple predicting algorithms or estimators and the resulting predictions. It is a boost of the weights of the training sets of the estimators from multiple or combined sources. The methodology in AdaBoost sequentially fits weak learners modified data repeatedly, the combination of which produces the final prediction. This technique reduces the bias of the estimators thereby producing a better output.

Create the Easy Ensemble and AdaBoost model, train and evaluate

```python
# Easy ensenble classifier and AdaBoost classifier
from sklearn.ensemble import AdaBoostClassifier
base_estimator = AdaBoostClassifier(n_estimators=25)

from imblearn.ensemble import EasyEnsembleClassifier
easy_ensemble = EasyEnsembleClassifier(n_estimators=25, base_estimator=base_estimator)

easy_ensemble.fit(X_train, y_train)

y_pred = easy_ensemble.predict(X_test)


print("Easy ensemble classifier performance:")
print(f"Balanced accuracy: {balanced_accuracy_score(y_test, y_pred):.3f} - ")
```

https://towardsdatascience.com/machine-learning-workflow-on-diabetes-data-part-01-573864fcc6b8

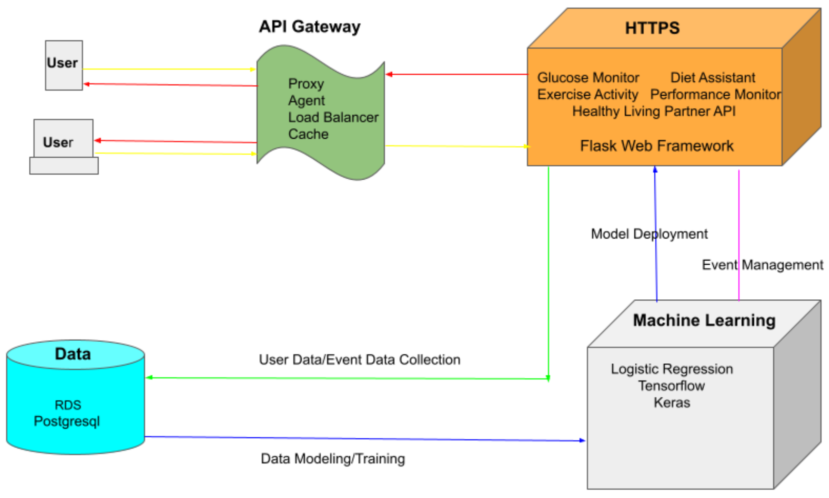


Figure-5 Healthy Living Partner System Diagram

Key Features & Scope

The proposed solution is an app that assists people to live a healthy life through monitored and controlled practice, thereby preventing chronic disease. Healthy Living Partner provides guides and recommendations to users as they approach and go through daily diet, exercise and health monitoring routine. It is the go-to-place for habit and lifestyle formation and modelling.

Product Features

1. Health Status Check
2. Diet Assistant
3. Exercise Activity Unit
4. Glucose Monitor
5. Performance Monitor
6. Application Programming Interface(API)

- Health Status Check (Health Status)

Provides analytical report of user health condition

- Diet Assistant (Diet List)

Provides a list of diet menu with recommended options for Breakfast, Lunch and Dinner

- Exercise Activity Unit (Exercise List)

Provides the user with recommended exercise activity

- Glucose/Sugar-Level Monitor (Glucose Monitor)

Provides the user with measures of Blood Glucose level

- Performance Monitor (Performance Monitor)

Provides the user with an analysis and percentage rating of completion of tasks in a routine.

- Application Programming Interface (API)

This feature is a technical development component purposely for integration with other complimentary products. API documentation is provided to developers upon request.

Data collection and usage in HLP

1. The Health Status Check feature checks the health status using the available data on the user's health. It determines whether the user is healthy or unhealthy using a baseline threshold. Users are provided with diet and exercise recommendations.
2. The Performance Monitor gathers data relating to any aspect of the application that is used based on the events. For example, if the user adheres to diet and exercise routine recommended, activity is tracked, monitored and rated. This is to determine how well or closely the recommendation is being followed. It also serves as a measure of the user's commitment and accountability.
3. Data is collected in one form or the other by all feature components of the system.

Data may be collected and utilized within the application to enhance the intelligence and feature capabilities as needed.

Data Storage

Data gathered and/or collected by the application is stored or warehoused in the cloud through cloud services. This is an essential part of the concept for scalability purposes.

Prototype

Proof of concept is provided with some pages of the apps as shown below along with a [link](#).

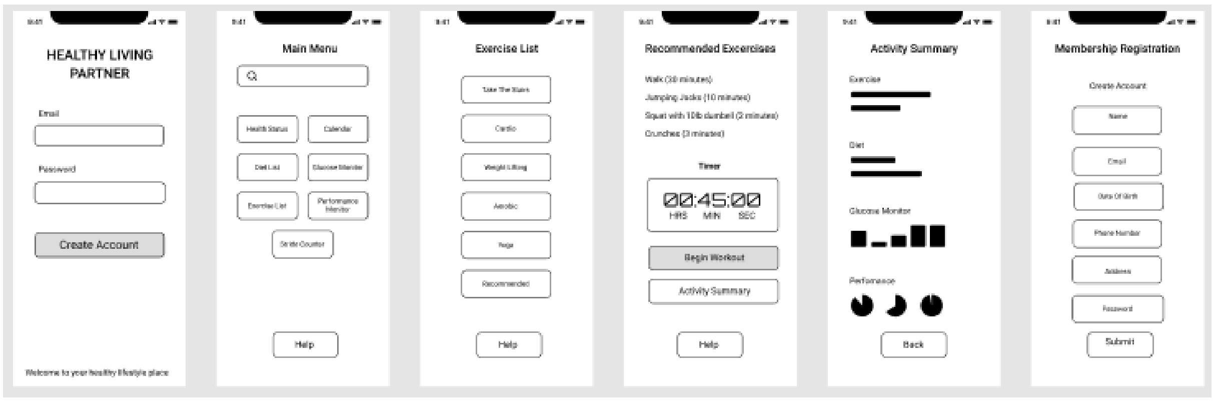


Fig-4 Pages (Healthy Living Partner)

[Healthy Living Partner(HLP)](#)

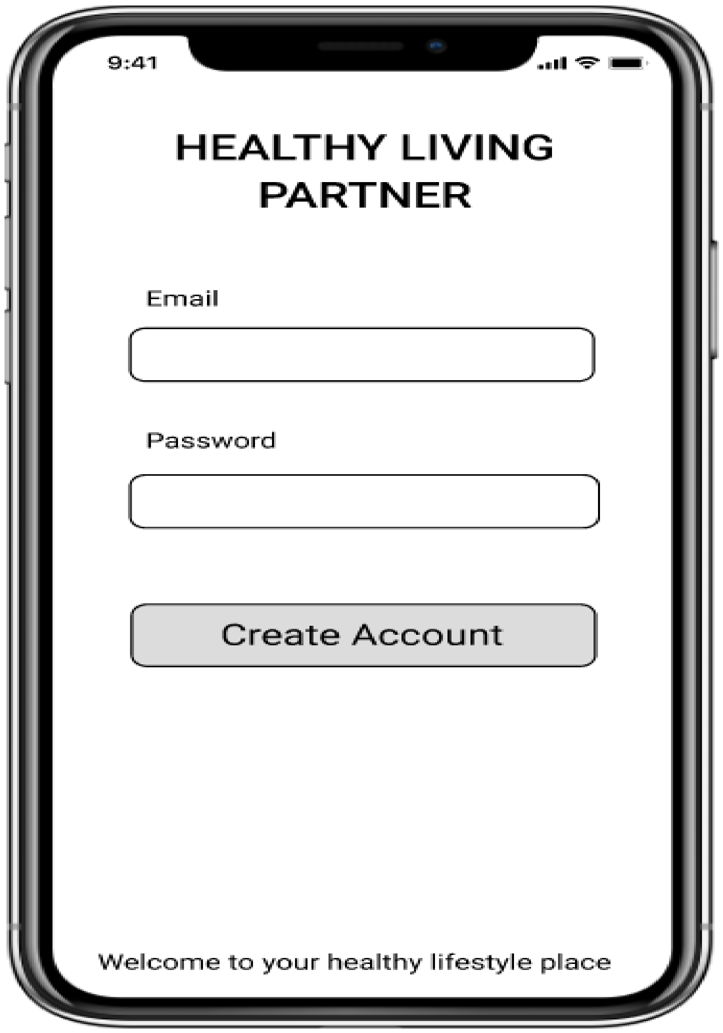


Figure-5 Healthy Living Partner

Refinement

The Machine Learning techniques applied in this proposal are mainly focused on proof of design concepts and principles. One can refine the system by expanding on the following:

1. Adjust the hyperparameters for the algorithms for optimization
   Note that the hyperparameters as well as the effects are different for each one of the algorithms and should be observed and documented.

2. Create automation for the process. In other words, as we collect more data for analysis, the result is fed into the decision support system automatically.

3. Create visualization for the performance result for each algorithm in the experiment.

Results

Model Evaluation and Validation

Given the circumstances and the conditions surrounding our setup in this experiment, the **Easy Ensemble Classifier** produced a balanced accuracy score of **0.884** or **88.4%**. This accuracy score meets and exceeds the minimum set requirement of 70%.

What is more encouraging about the performance is the potential for improved performance, if we have more data and more iterations along with more processing power. The algorithms demonstrated the capability to learn better over so many iterations.

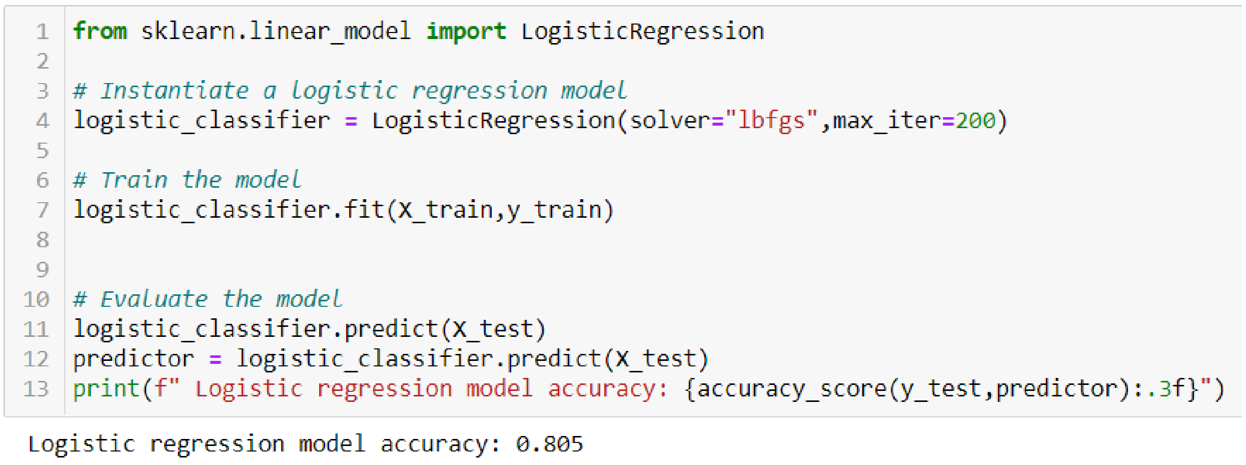
Justifying the set benchmarks, the **Logistic Regression** model performed well at **0.805** or **80.5%**, which is in line with the requirement of this experiment. It provides validation for its simplicity and efficiency as proven in this experiment and other similar experiments. If we narrow down to feature selection, we can investigate the features with the most effect on the outcome and get the best possible performance out of the algorithm. As a result, this will be the preferred and chosen model for the solution.

The **Neural Network** model, **TensorFlow/Keras** did not disappoint with its output **0.708** or **70.8%**. Low accuracy score is partly due to small size data and limited iterations of 100 epochs. With higher and better computational resources, the output for this model can be improved significantly.

We should note that the result only identifies the Easy Ensemble Classifier as the best performer in this particular experiment out of all the models applied. The result could be different if the circumstances or factors or parameters change. Determining the best classifier depends on so many factors and it is a much broader scope beyond this basic experiment.

Here is how each individual algorithm performed:

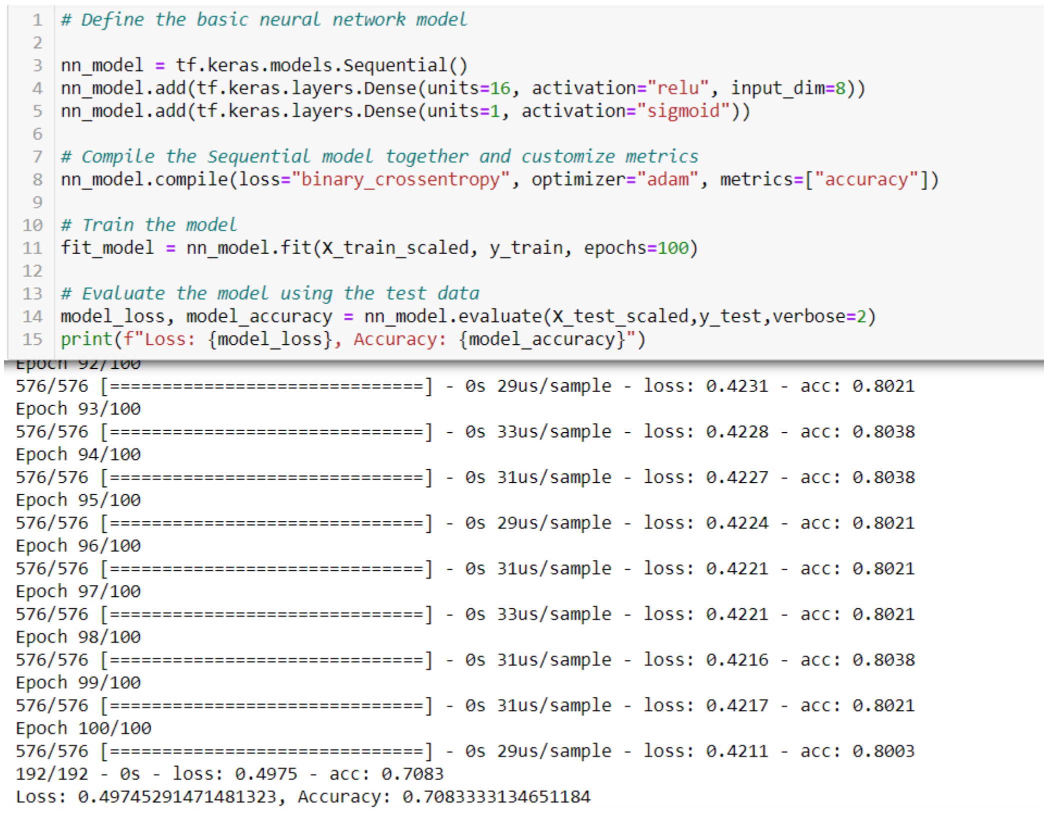
Logistic Regression



```python
from sklearn.linear_model import LogisticRegression

# Instantiate a logistic regression model
logistic_classifier = LogisticRegression(solver="lbfgs",max_iter=200)

# Train the model
logistic_classifier.fit(X_train,y_train)


# Evaluate the model
logistic_classifier.predict(X_test)
predictor = logistic_classifier.predict(X_test)
print(f" Logistic regression model accuracy: {accuracy_score(y_test,predictor):.3f}")
```

```
 Logistic regression model accuracy: 0.805
```

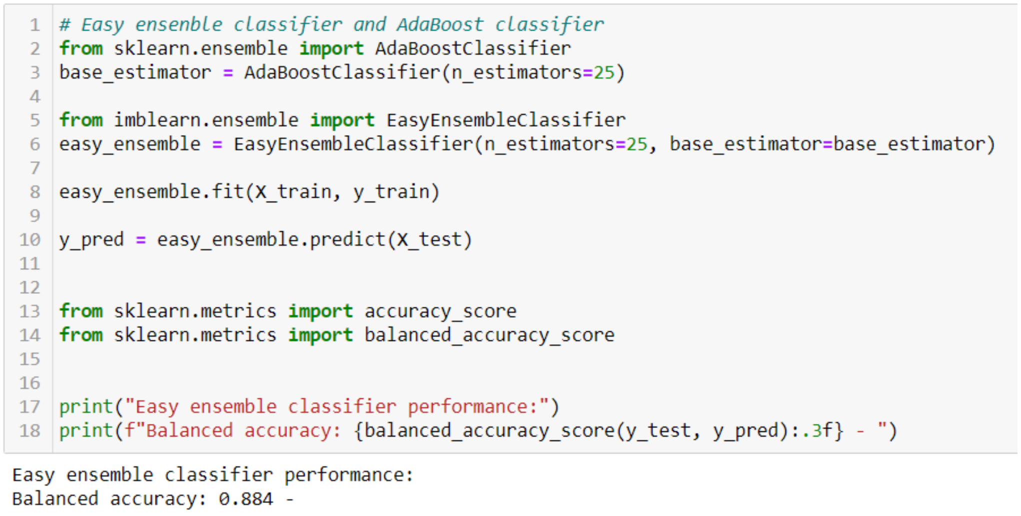
Neural Network



```python
# Define the basic neural network model

nn_model = tf.keras.models.Sequential()
nn_model.add(tf.keras.layers.Dense(units=16, activation="relu", input_dim=8))
nn_model.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Compile the Sequential model together and customize metrics
nn_model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])

# Train the model
fit_model = nn_model.fit(X_train_scaled, y_train, epochs=100)

# Evaluate the model using the test data
model_loss, model_accuracy = nn_model.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
Epoch 92/100
576/576 [==============================] - 0s 29us/sample - loss: 0.4231 - acc: 0.8021
Epoch 93/100
576/576 [==============================] - 0s 33us/sample - loss: 0.4228 - acc: 0.8038
Epoch 94/100
576/576 [==============================] - 0s 31us/sample - loss: 0.4227 - acc: 0.8038
Epoch 95/100
576/576 [==============================] - 0s 29us/sample - loss: 0.4224 - acc: 0.8021
Epoch 96/100
576/576 [==============================] - 0s 31us/sample - loss: 0.4221 - acc: 0.8021
Epoch 97/100
576/576 [==============================] - 0s 33us/sample - loss: 0.4221 - acc: 0.8021
Epoch 98/100
576/576 [==============================] - 0s 31us/sample - loss: 0.4216 - acc: 0.8038
Epoch 99/100
576/576 [==============================] - 0s 31us/sample - loss: 0.4217 - acc: 0.8021
Epoch 100/100
576/576 [==============================] - 0s 29us/sample - loss: 0.4211 - acc: 0.8003
192/192 - 0s - loss: 0.4975 - acc: 0.7083
Loss: 0.49745291471481323, Accuracy: 0.7083333134651184
```

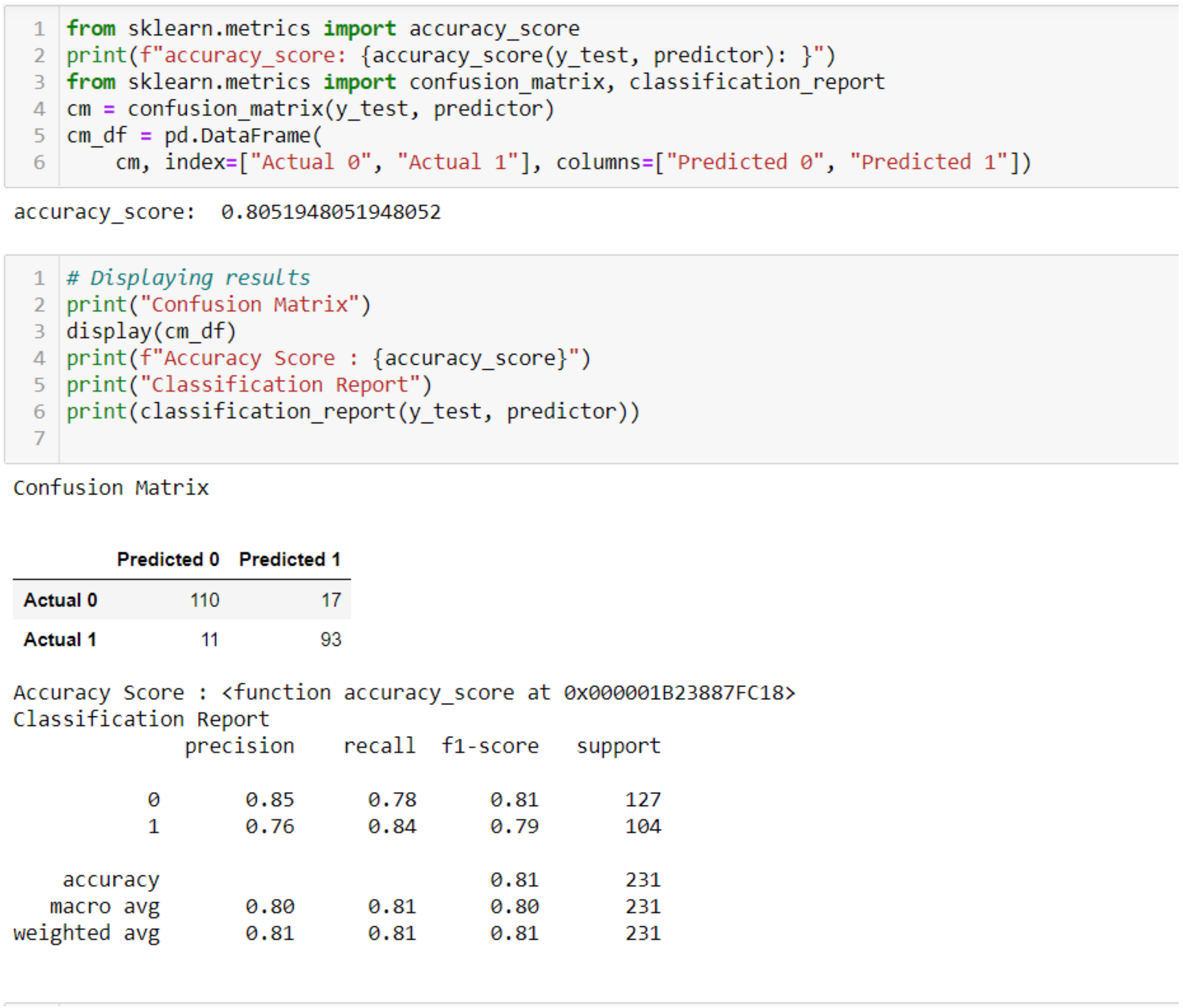
Easy Ensemble and AdaBoost Classifier



```python
# Easy ensemble classifier and AdaBoost classifier
from sklearn.ensemble import AdaBoostClassifier
base_estimator = AdaBoostClassifier(n_estimators=25)

from imblearn.ensemble import EasyEnsembleClassifier
easy_ensemble = EasyEnsembleClassifier(n_estimators=25, base_estimator=base_estimator)

easy_ensemble.fit(X_train, y_train)

y_pred = easy_ensemble.predict(X_test)


from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score


print("Easy ensemble classifier performance:")
print(f"Balanced accuracy: {balanced_accuracy_score(y_test, y_pred):.3f} - ")
```

```
Easy ensemble classifier performance:
Balanced accuracy: 0.884 -
```

The Accuracy Score and Confusion Matrix



```python
from sklearn.metrics import accuracy_score
print(f"accuracy_score: {accuracy_score(y_test, predictor): }")
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, predictor)
cm_df = pd.DataFrame(
    cm, index=["Actual 0", "Actual 1"], columns=["Predicted 0", "Predicted 1"])
```

```
accuracy_score:  0.8051948051948052
```

```python
# Displaying results
print("Confusion Matrix")
display(cm_df)
print(f"Accuracy Score : {accuracy_score}")
print("Classification Report")
print(classification_report(y_test, predictor))

```

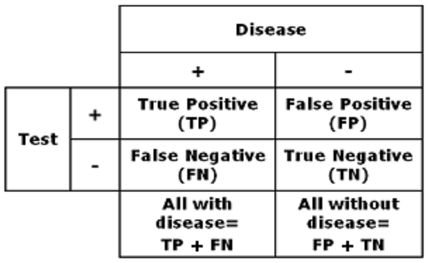
```
Confusion Matrix
```

	Predicted 0	Predicted 1
Actual 0	110	17
Actual 1	11	93

```
Accuracy Score : <function accuracy_score at 0x000001B23887FC18>
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.78      0.81       127
           1       0.76      0.84      0.79       104

    accuracy                           0.81       231
   macro avg       0.80      0.81      0.80       231
weighted avg       0.81      0.81      0.81       231
```

The accuracy score is simply the percentage of predictions that are correct. In this case, the model's accuracy score was 0.810, meaning that the model was correct 81% of the time.

Metrics Evaluated



		Disease	
		+	-
Test	+	True Positive (TP)	False Positive (FP)
	-	False Negative (FN)	True Negative (TN)
		All with disease= TP + FN	All without disease= FP + TN

Fig-6 Showing TP/FP/FN/TN Relationship (Image Source)

```
# Display metrics evaluated

metrics = evaluate(predictor, X_test, y_test, True)
```

```
Predictions  0.01
Actuals
0             125
1              67

Recall:     0.500
Precision:  0.349
Accuracy:   0.500

True Positive Rate(TP): 67.000
False Positive Rate(FP): 125.000
False Negative Rate(FN): 67.000
True Negative Rate(TN): 125.000
```

Classification Report - imbalanced

```
# Print the imbalanced classification report
from imblearn.metrics import classification_report_imbalanced


# Displaying results
print("Confusion Matrix")
display(cm_df)
print(f"Accuracy Score : {acc_score}")
print(f"Classification Report:Easy Ensemble & AdaBoost Classifier")

print(classification_report_imbalanced(y_test, y_pred))
```

```
Confusion Matrix
```

	Predicted 0	Predicted 1
Actual 0	105	20
Actual 1	32	35

```
Accuracy Score : 0.745134328358209
Classification Report:Easy Ensemble & AdaBoost Classifier
                   pre       rec       spe        f1       geo       iba       sup

          0       0.86      0.76      0.76      0.81      0.76      0.58       125
          1       0.63      0.76      0.76      0.69      0.76      0.58        67

avg / total       0.78      0.76      0.76      0.76      0.76      0.58       192
```

ROC_AUC Curve

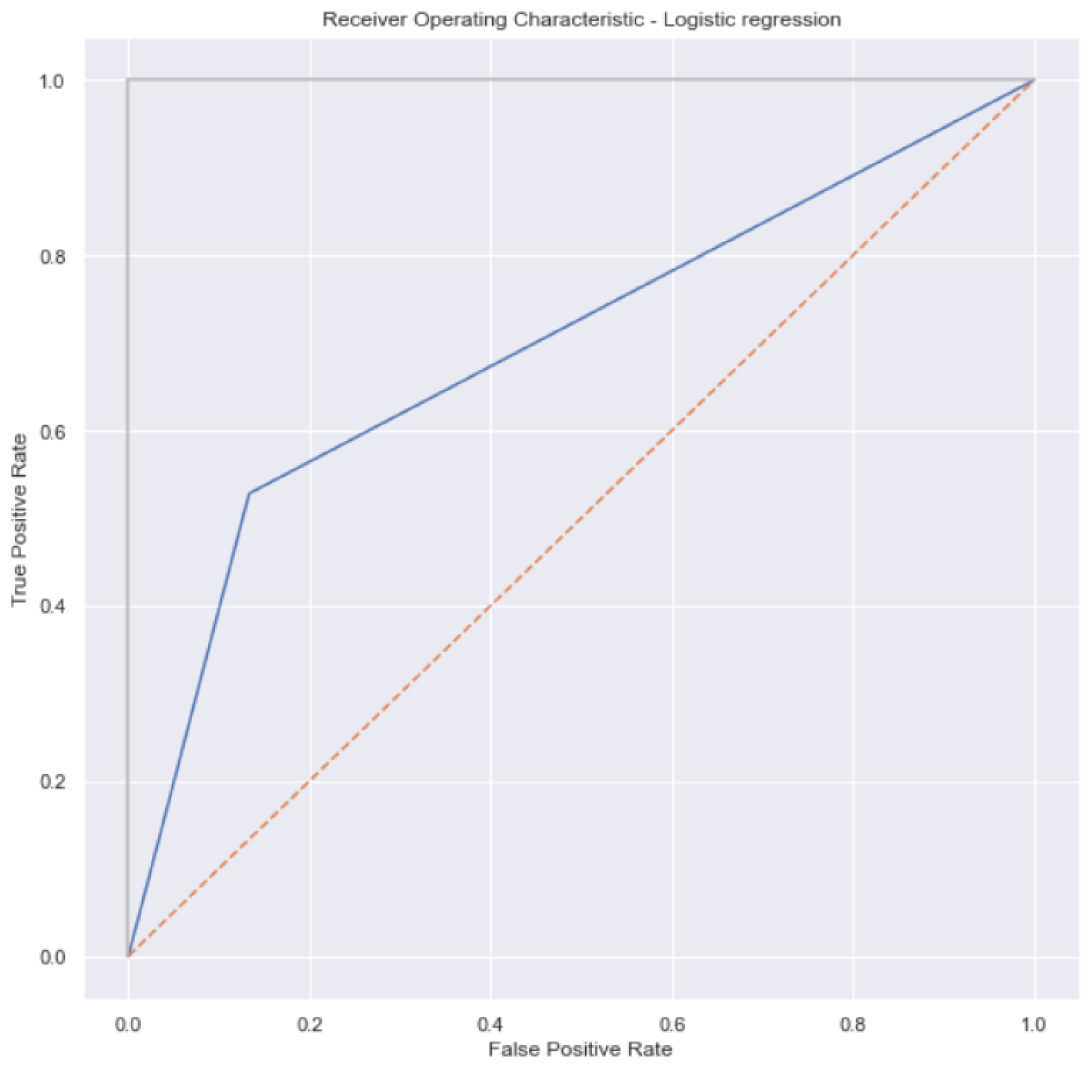
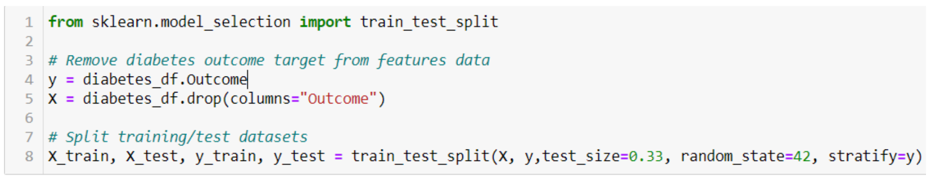


Fig-6 ROC Curve

Justification

Data is expected to grow significantly as more users sign on and use the app more frequently. In anticipation of the Big Data scenario in a real production environment, the system will need to apply a set of robust algorithms for higher efficacy.

Splitting the data for training and testing



```python
from sklearn.model_selection import train_test_split

# Remove diabetes outcome target from features data
y = diabetes_df.Outcome
X = diabetes_df.drop(columns="Outcome")

# Split training/test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.33, random_state=42, stratify=y)
```

It is important to set a portion of the dataset for training and testing. **Training dataset** is used to learn the pattern. **Testing dataset** is used to assess the performance. If the entire dataset is used to train the model, you won't know how well the model will perform when it encounters new data. X is the input and y is the prediction (i.e output).

test_size - represents the proportion of dataset to include in the test split
random_state - controls the shuffling of the data before the split
stratify - data is split in a stratify fashion, otherwise samples will be assigned randomly using it as class labels.

- Setting the **solver** at "lbfgs" and the **max_iter** at 200, the **Logistic Regression** model produced an accuracy of 0.805 or 80.5%.

- The **neural network (TensorFlow/Keras)** showed an accuracy of 0.708 or 70.8% with a loss of 0.497 or 49.7%. This is due largely to the choice of **units**, **activation** and **epochs**. The result will be much improved in an environment with higher resource capacity running higher epochs.

- The Easy Ensemble and AdaBoost model gave a Balanced Accuracy 0.884 or 88.4%. AdaBoost is the primary or base algorithm, along with other algorithms combined to produce an output. Changing the value of the **n_estimator** parameter or any of the **combined algorithms** will affect the outcome.

Conclusion

Reflection

The overall solution objective is improved health and wellness through controlled diet and exercise activities, paving way for reduced costs of diagnosing and treating chronic diseases. Healthy Living Partner(HLP) offers this solution with its core features namely,Health Status Check, Glucose Monitor, Diet Assistant, Exercise Activity Unit, Performance Monitor and API.

Machine Learning algorithm is the main driver behind decisions made in terms of what actions to take, as well as, recommend to the user. It collects and analyzes data generated by the user for guidance on maintaining a healthy lifestyle. For these reasons it is absolutely necessary for the Machine learning algorithm to continue to perfect itself in learning, training and making predictions in support of decisions.

As articulated and demonstrated in this proposal, Logistic Regression, Neural Network (TensorFlow/Keras) and Easy Ensemble AdaBoost algorithms were used to demonstrate the capabilities of the application of machine learning in this solution to determine who's healthy and who's not healthy. Further, output trends indicate improving or deteriorating health conditions. Overall, the Logistic Regression offer the best fit for the solution.

The most challenging aspect of the design is the automation of the cycle of collecting data, analyzing the data, train/test to identify patterns and make predictions which is then incorporated into the system to enhance the decision process. Also, adding performance tuning to the cycle as all in one automated system

Improvement

1. The size of the data for this experiment is very small. A larger size of data should produce a more reliable output.

2. The data should not be limited to a particular segment of the population(as in the case of pregnant women).

3. Expanding the API coverage
   There are other health related devices that monitor other conditions in humans, especially those with pre-existing conditions. Expanding the API coverage to other test, reporting and monitoring devices for critical data collection and analysis to support decisions.

4. Biases in the data
   It is highly recommended to make provision that will address any anticipated and occuring biases in the dataset.

5. Multiple decision criteria algorithms
   Investigate the use of algorithms that utilize multiple decision criterias for improved accuracy.

6. The deployment phase
   Create and develop the python script for deployment

References

1. Accuracy
https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b
2. Pima Indians Diabetic Data
https://www.kaggle.com/uciml/pima-indians-diabetes-database
3. Economic Costs of Diabetes in the US 2017
https://care.diabetesjournals.org/content/early/2018/03/20/dci18-0007
4. Healthcare statistics for 2021
https://policyadvice.net/insurance/insights/healthcare-statistics
5. Complications of Type1 Diabetes
https://jdrf.org.uk/information-support/about-type-1-diabetes/complications/
6. Predictive Models for Diabetes Mellitus Using machine Learning

https://doi.org/10.1186/s12902-019-0436-6

7. Balanced and Imbalanced Dataset
https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5#:~:text=Balance%20Dataset,positive%20values%20and%20negative%20values.

8. The Logistic Regression Theory
https://www.pugetsystems.com/labs/hpc/Machine-Learning-and-Data-Science-Logistic-Regression-Theory-988/#:~:text=Logistic%20Regression%20the%20Theory&text=Logistic%20regression%20makes%20use%20of,%2C%20good%20or%20bad%20etc..

9. Neural Network Elements
https://wiki.pathmind.com/neural-network

10. Ensemble Methods
https://scikit-learn.org/stable/modules/ensemble.html