Diabetes Prediction ML

AHMED ALI

MS-23-081@PIEAS.EDU.PK

SUPERVISED BY:

PROF. DR. SHAHZAD AHMED QURESHI

Introduction

Diabetes is a common chronic disease that can be dangerous. Diabetes can be identified when blood glucose is higher than normal level, which is caused by high secretion of insulin or biological effects. Diabetes can cause various damage to our body and can disfunction tissues, kidneys, eyes and blood vessels.

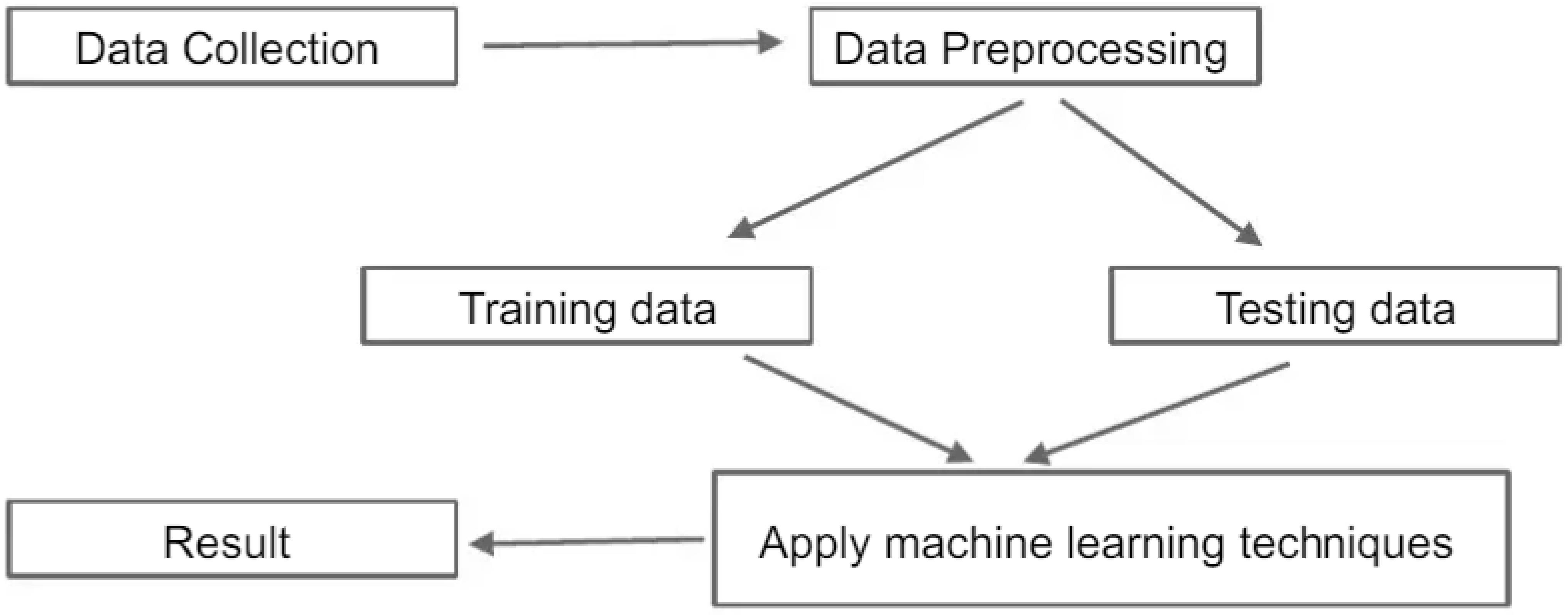
Introduction

Diabetes can be divided into two categories, type 1 diabetes and type 2 diabetes. Patients with type 1 diabetes are normally younger with an age less then 30 years old. The clinical symptoms are increase thirst and frequent urination this type of diabetes cannot be cleared by medications as it requires therapy. Type 2 diabetes occurs more commonly on middle-aged and old people, which can show hypertension, obesity and other diseases. with our living standards diabetes has increased commonly in people’s daily life. So how to analyze diabetes is worth studying.

Problem Statement

Diabetes is a most common disease caused by a group of metabolic disorders. It is also known as Diabetic mellitus. It affects the organs of the human body. It can be controlled by predicting this disease earlier. If diabetics patient is untreated for a long time, it may lead to increase blood sugar. Now a days, Healthcare industries generating large volume of data. Machine Learning algorithms and statistics are used to predict the disease with the help of current and past data. Machine learning techniques helps the doctors to predict early stage for diabetics. Diabetics patient medical record and different types of algorithms are added in dataset for experimental analysis.

Block Diagram/Flowchart:

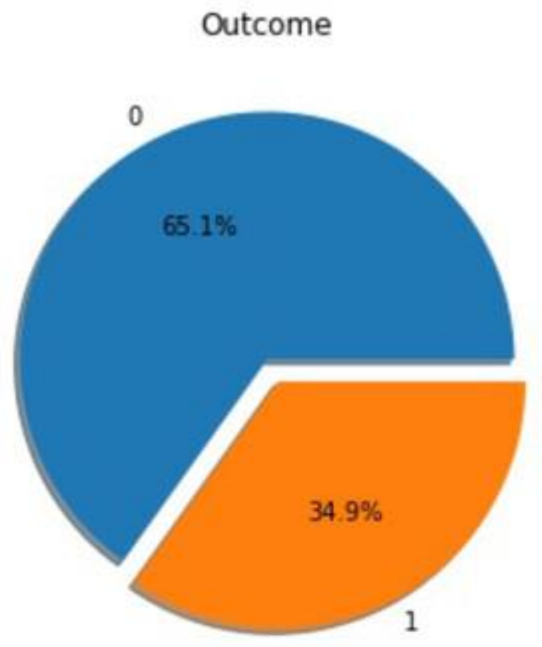


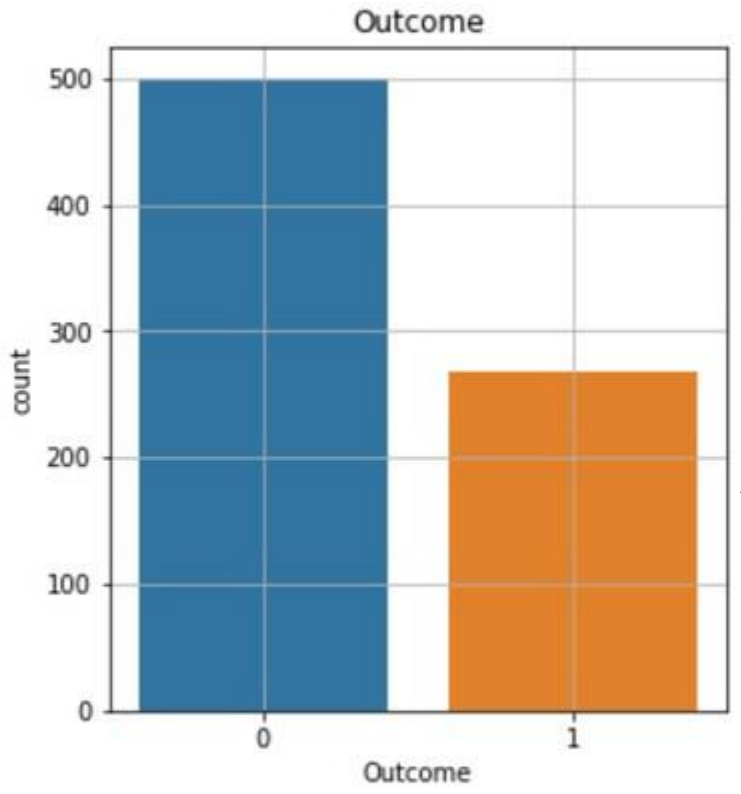
Dataset

The Pima Indian dataset is an open-source dataset [6] that is publicly available for machine learning classification, which has been used in this work along with a private dataset. It contains 768 patients’ data, and 268 of them have developed diabetes.

Percentage of people having diabetes in the Pima Indian dataset.

Diabetes prediction using machine learning and explainable AI techniques



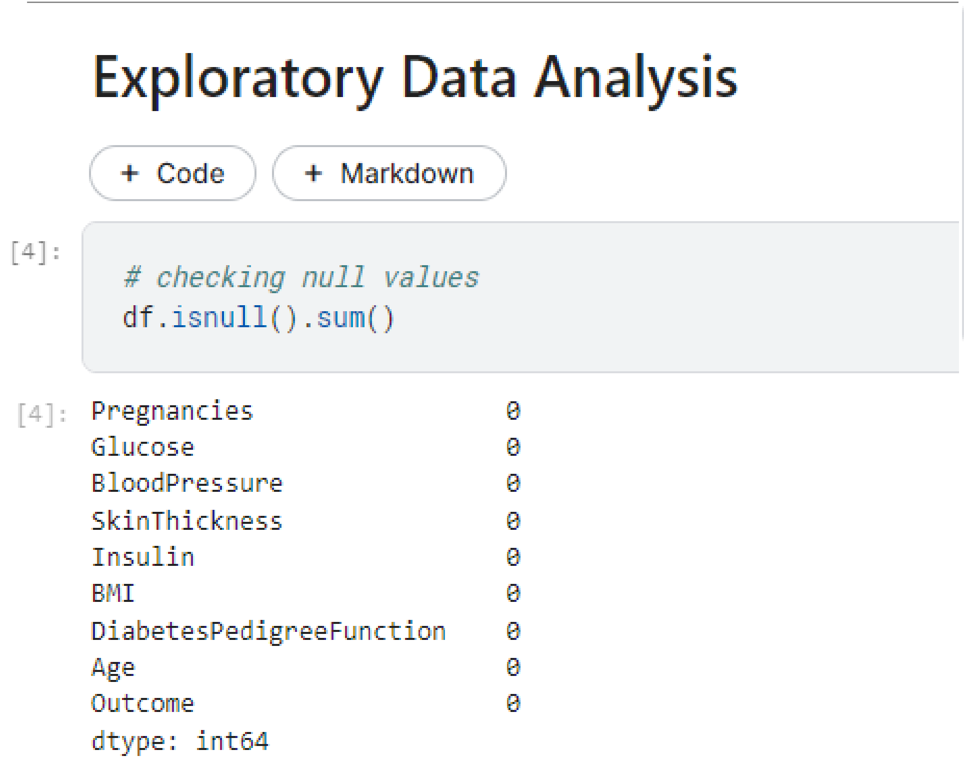


Healthcare Tech Letters, Volume: 10, Issue: 1-2, Pages: 1-10, First published: 14 December 2022, DOI: (10.1049/htl2.12039)

Features of the Pima Indian Dataset

Pregnancies	**Skin thickness**	**Diabetes pedigree function**
Glucose	Insulin	Age
Blood pressure	BMI	

Data Analysis

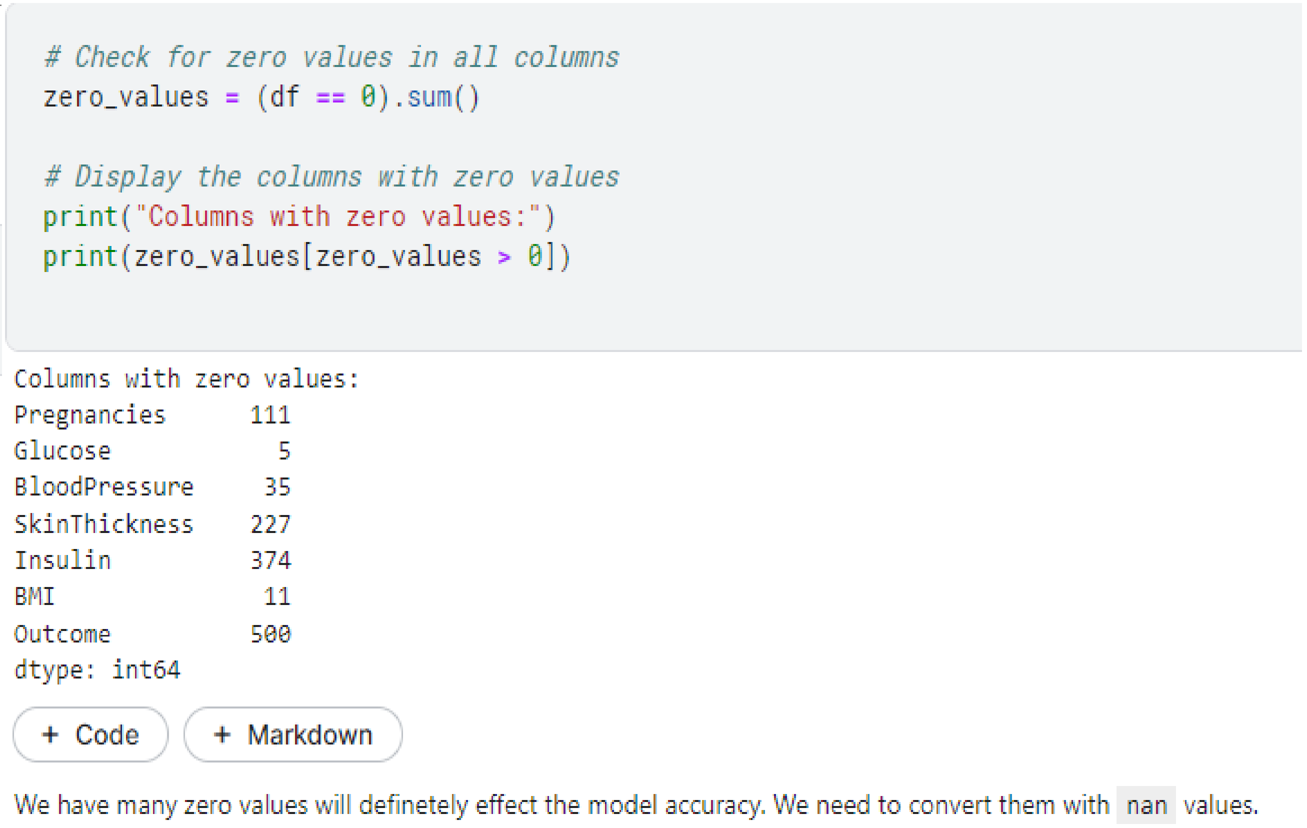


Exploratory Data Analysis

+ Code + Markdown

```
[4]:
# checking null values
df.isnull().sum()
```

```
[4]: Pregnancies                 0
     Glucose                     0
     BloodPressure               0
     SkinThickness               0
     Insulin                     0
     BMI                         0
     DiabetesPedigreeFunction    0
     Age                         0
     Outcome                     0
     dtype: int64
```



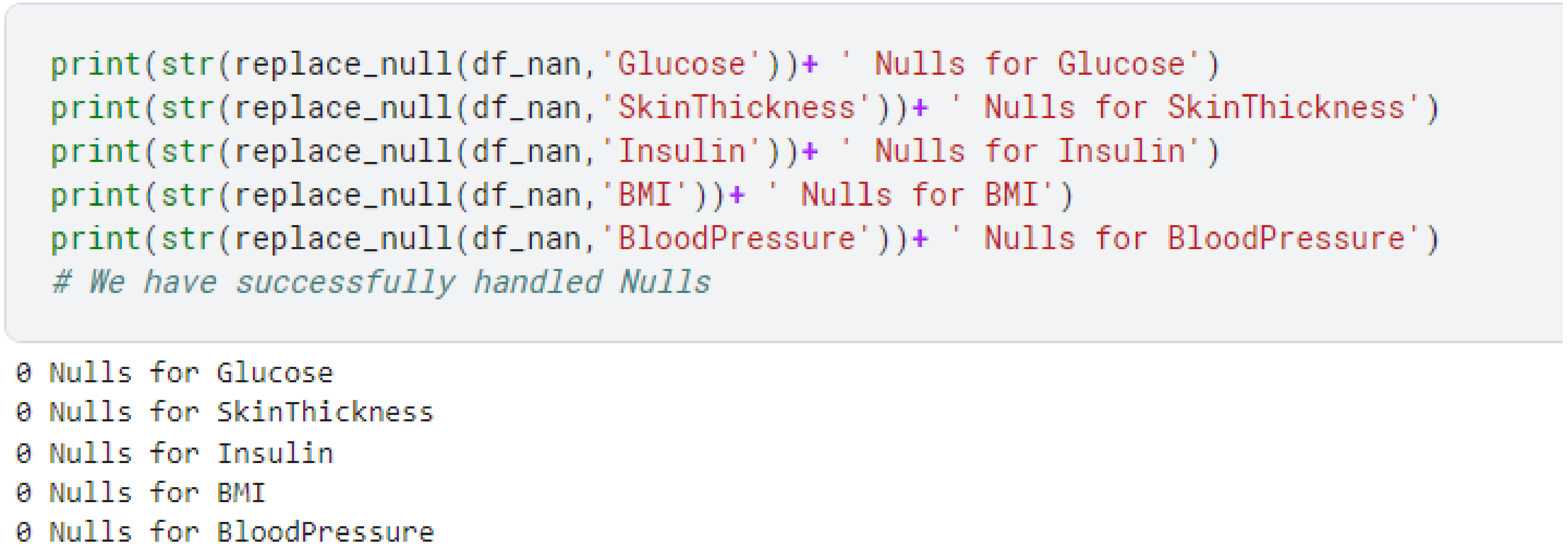
```python
# Check for zero values in all columns
zero_values = (df == 0).sum()

# Display the columns with zero values
print("Columns with zero values:")
print(zero_values[zero_values > 0])
```

```
Columns with zero values:
Pregnancies      111
Glucose            5
BloodPressure     35
SkinThickness    227
Insulin          374
BMI               11
Outcome          500
dtype: int64
```

+ Code + Markdown

We have many zero values will definetely effect the model accuracy. We need to convert them with `nan` values.

Data Analysis

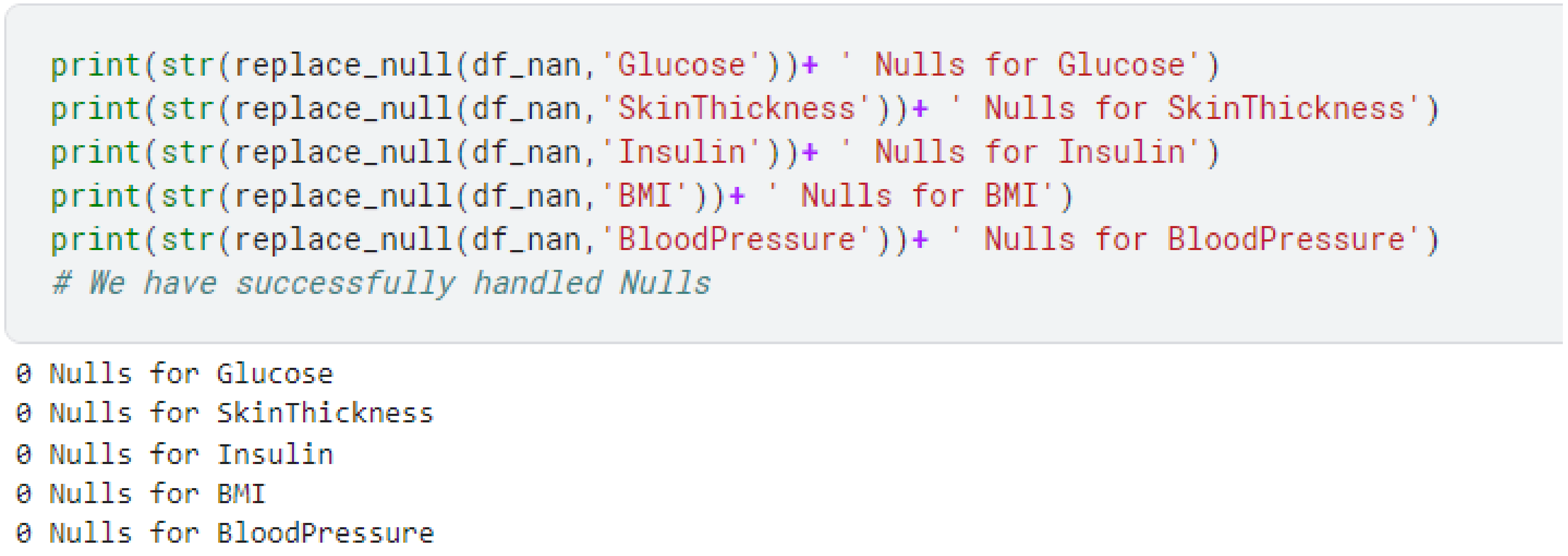


```
print(str(replace_null(df_nan,'Glucose'))+ ' Nulls for Glucose')
print(str(replace_null(df_nan,'SkinThickness'))+ ' Nulls for SkinThickness')
print(str(replace_null(df_nan,'Insulin'))+ ' Nulls for Insulin')
print(str(replace_null(df_nan,'BMI'))+ ' Nulls for BMI')
print(str(replace_null(df_nan,'BloodPressure'))+ ' Nulls for BloodPressure')
# We have successfully handled Nulls
```

```
0 Nulls for Glucose
0 Nulls for SkinThickness
0 Nulls for Insulin
0 Nulls for BMI
0 Nulls for BloodPressure
```

All null values has been successfully imputed with their median.

Data Analysis

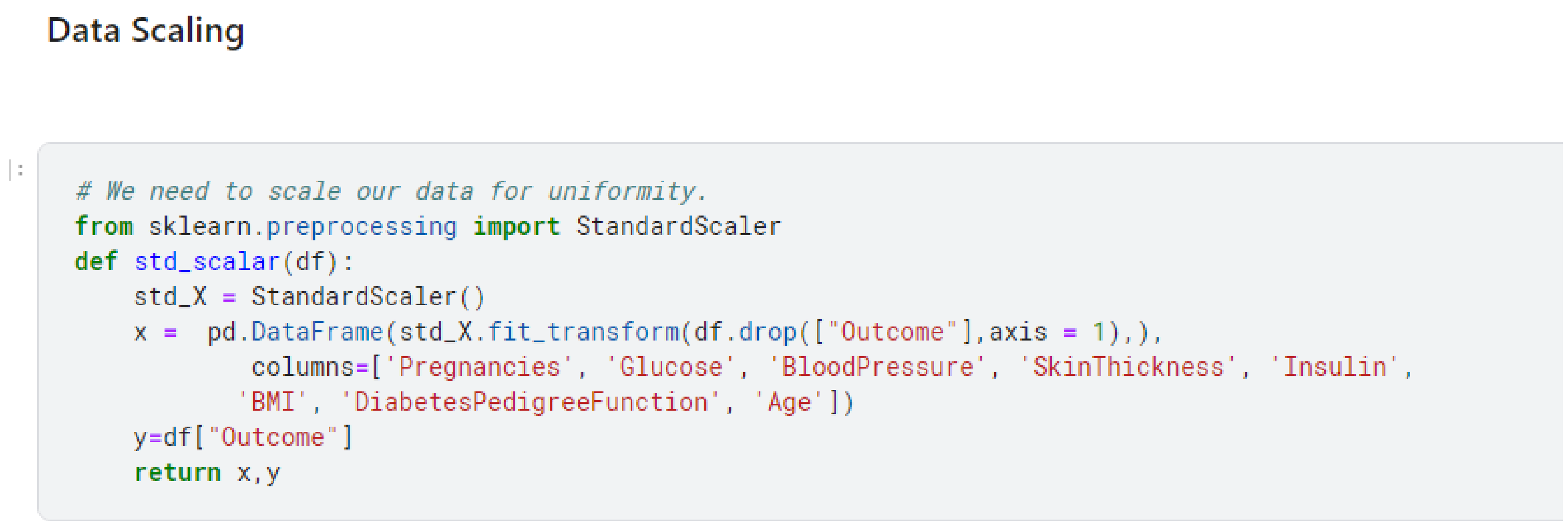


```python
print(str(replace_null(df_nan,'Glucose'))+ ' Nulls for Glucose')
print(str(replace_null(df_nan,'SkinThickness'))+ ' Nulls for SkinThickness')
print(str(replace_null(df_nan,'Insulin'))+ ' Nulls for Insulin')
print(str(replace_null(df_nan,'BMI'))+ ' Nulls for BMI')
print(str(replace_null(df_nan,'BloodPressure'))+ ' Nulls for BloodPressure')
# We have successfully handled Nulls
```

```
0 Nulls for Glucose
0 Nulls for SkinThickness
0 Nulls for Insulin
0 Nulls for BMI
0 Nulls for BloodPressure
```

All null values has been successfully imputed with their median.

Data Scaling

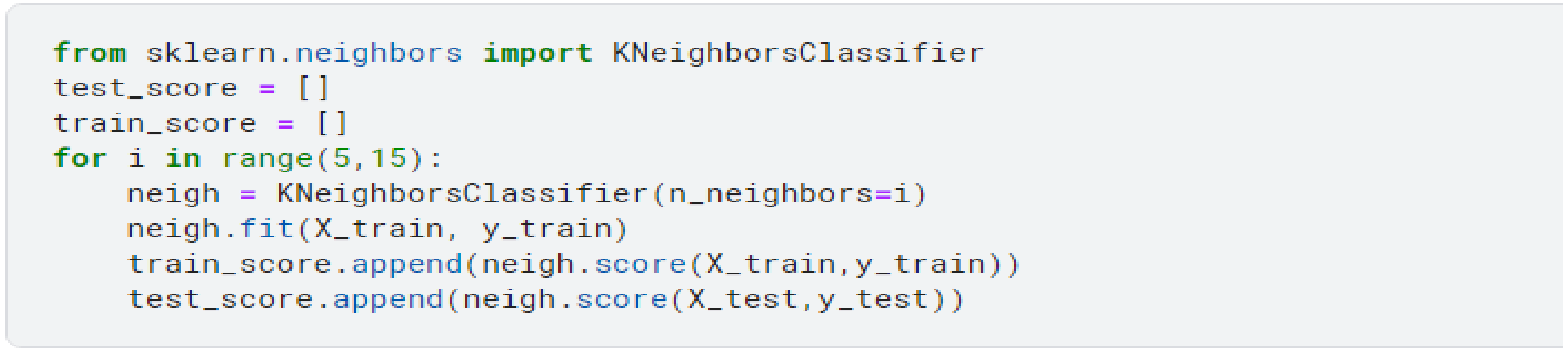


Data Scaling

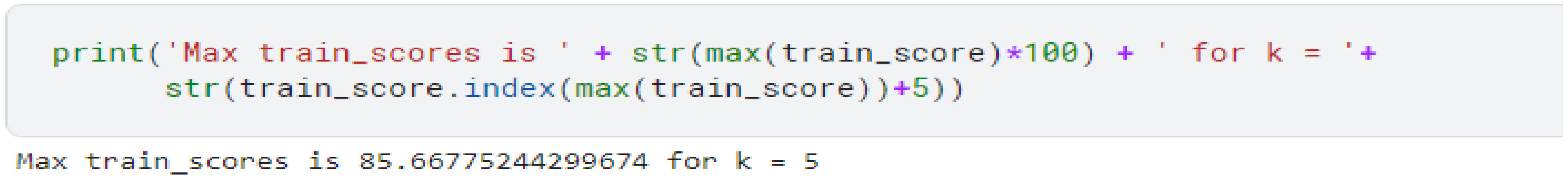
```
]:
# We need to scale our data for uniformity.
from sklearn.preprocessing import StandardScaler
def std_scalar(df):
    std_X = StandardScaler()
    x =  pd.DataFrame(std_X.fit_transform(df.drop(["Outcome"],axis = 1),),
            columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
           'BMI', 'DiabetesPedigreeFunction', 'Age'])
    y=df["Outcome"]
    return x,y
```

Model Training with KNN

Let's implement KNN



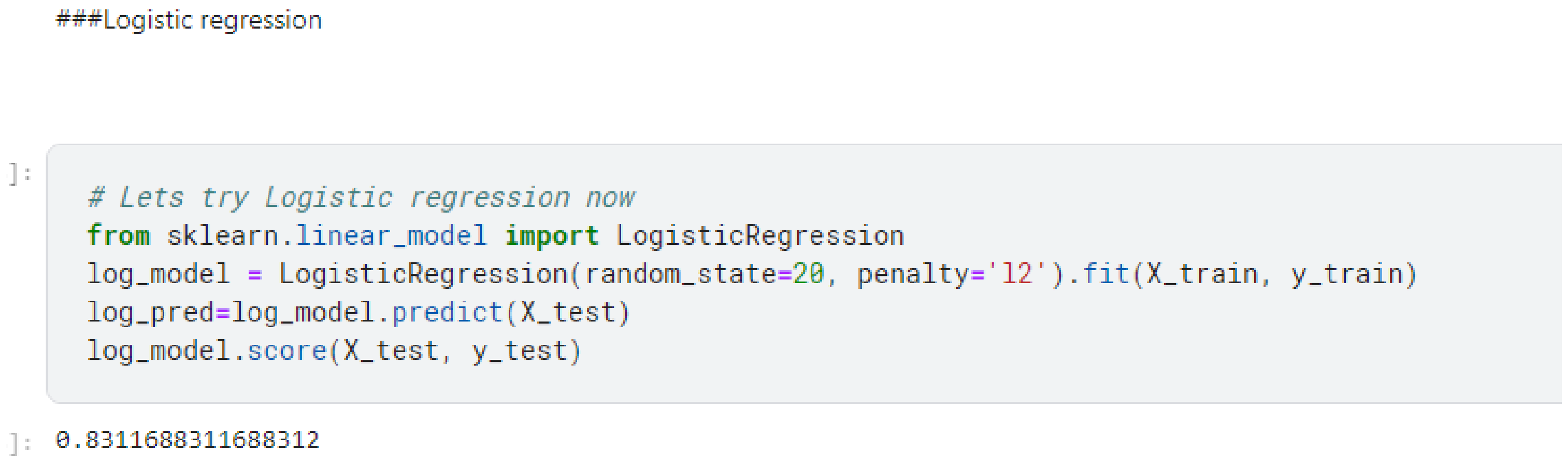
```
from sklearn.neighbors import KNeighborsClassifier
test_score = []
train_score = []
for i in range(5,15):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_train, y_train)
    train_score.append(neigh.score(X_train,y_train))
    test_score.append(neigh.score(X_test,y_test))
```



```
print('Max train_scores is ' + str(max(train_score)*100) + ' for k = '+
      str(train_score.index(max(train_score))+5))
```

```
Max train_scores is 85.66775244299674 for k = 5
```

Model Training with Logistic Regression

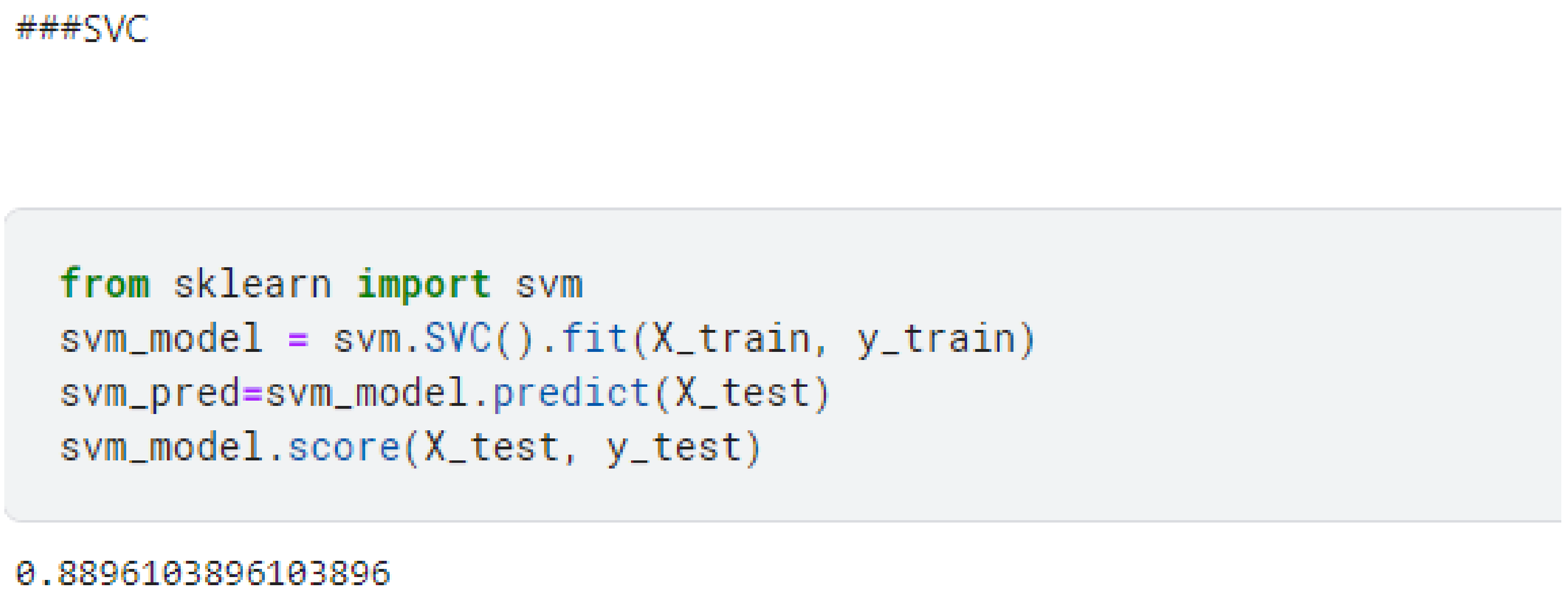


###Logistic regression

```
]:
# Lets try Logistic regression now
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression(random_state=20, penalty='l2').fit(X_train, y_train)
log_pred=log_model.predict(X_test)
log_model.score(X_test, y_test)
```

```
]: 0.8311688311688312
```

Model Training with SVC

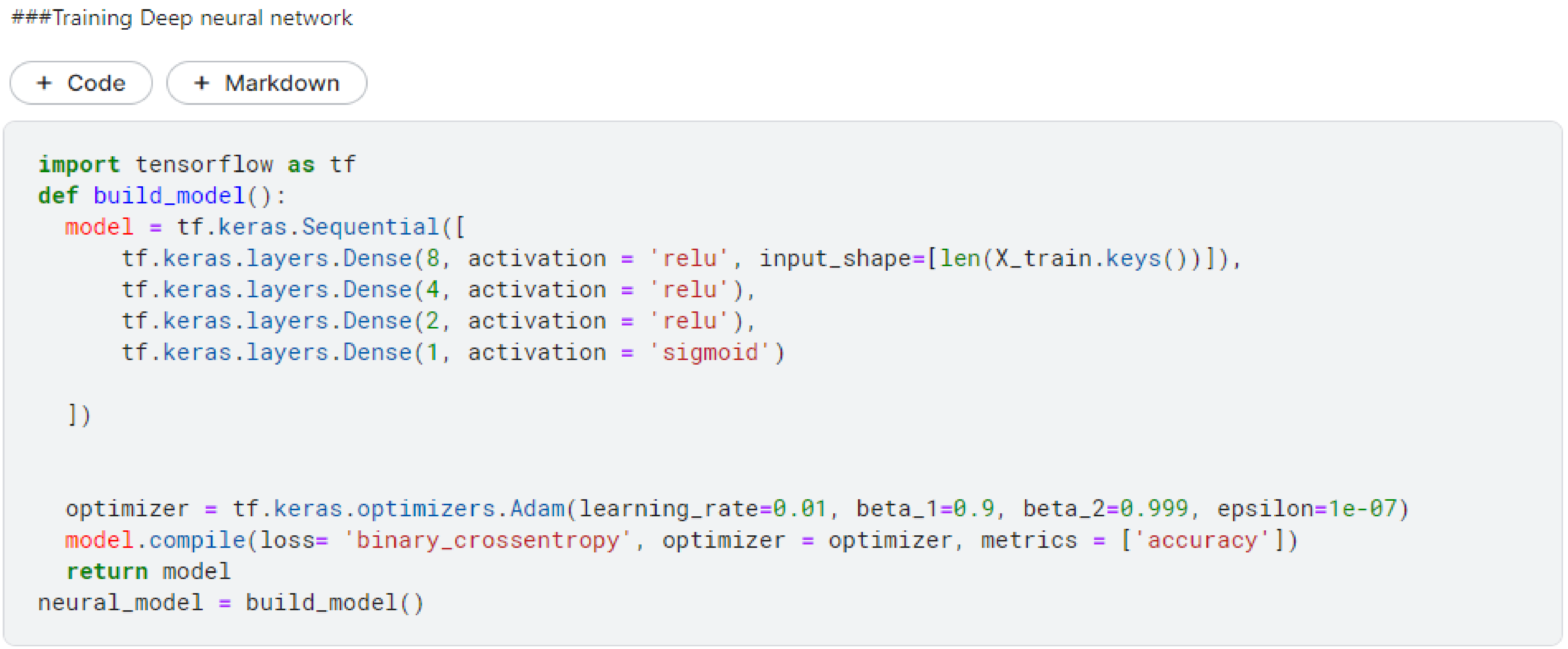


###SVC

```python
from sklearn import svm
svm_model = svm.SVC().fit(X_train, y_train)
svm_pred=svm_model.predict(X_test)
svm_model.score(X_test, y_test)
```

```
0.8896103896103896
```

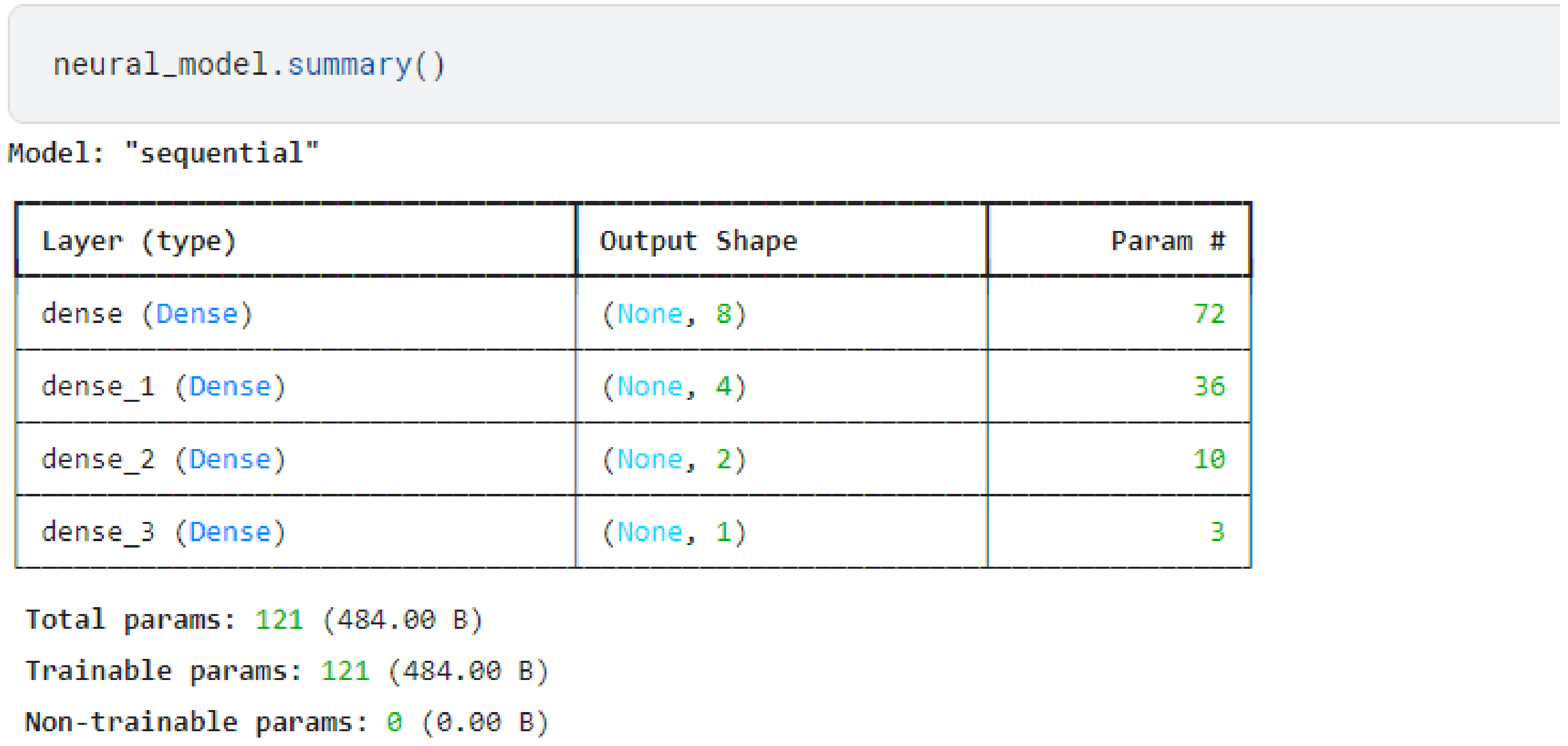
Model Training with Deep Neural Network



###Training Deep neural network

```
import tensorflow as tf
def build_model():
  model = tf.keras.Sequential([
      tf.keras.layers.Dense(8, activation = 'relu', input_shape=[len(X_train.keys())]),
      tf.keras.layers.Dense(4, activation = 'relu'),
      tf.keras.layers.Dense(2, activation = 'relu'),
      tf.keras.layers.Dense(1, activation = 'sigmoid')

  ])


  optimizer = tf.keras.optimizers.Adam(learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-07)
  model.compile(loss= 'binary_crossentropy', optimizer = optimizer, metrics = ['accuracy'])
  return model
neural_model = build_model()
```

Model Training with Deep Neural Network

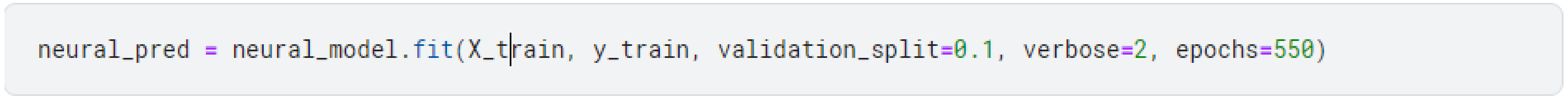


```python
neural_model.summary()
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense (Dense)                   │ (None, 8)              │            72 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 4)              │            36 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 2)              │            10 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                 │ (None, 1)              │             3 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 121 (484.00 B)

 Trainable params: 121 (484.00 B)

 Non-trainable params: 0 (0.00 B)
```

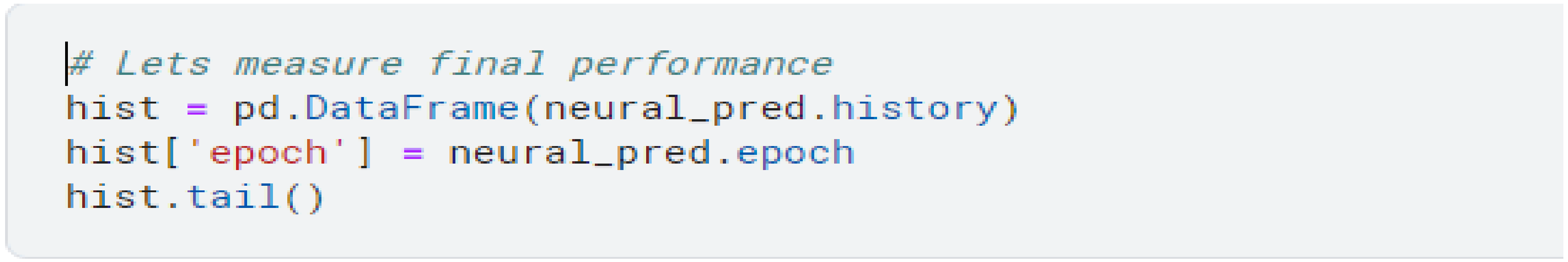
Fit Neural model on dataset.

Model Training with Deep Neural Network

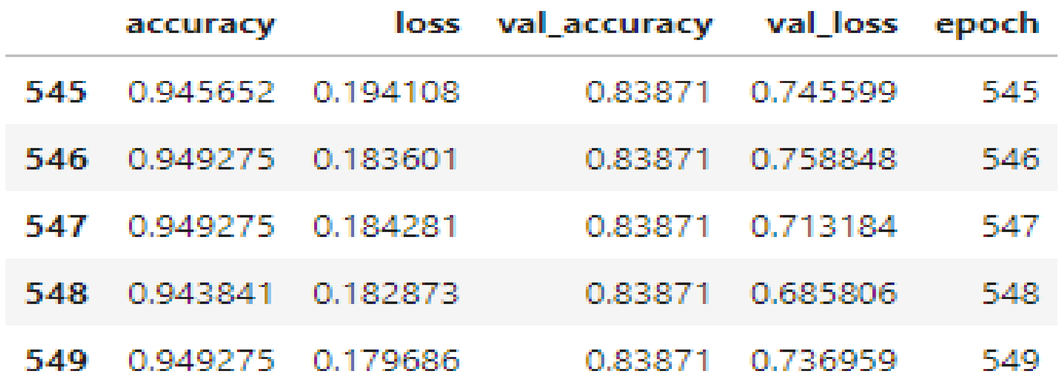


```
neural_pred = neural_model.fit(X_train, y_train, validation_split=0.1, verbose=2, epochs=550)
```

Model Training with Deep Neural Network

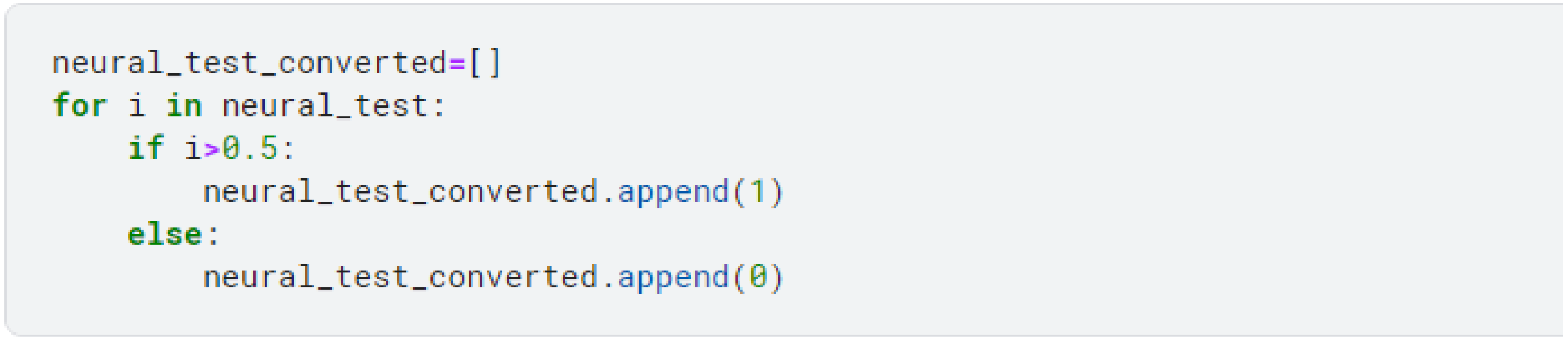


```
# Lets measure final performance
hist = pd.DataFrame(neural_pred.history)
hist['epoch'] = neural_pred.epoch
hist.tail()
```

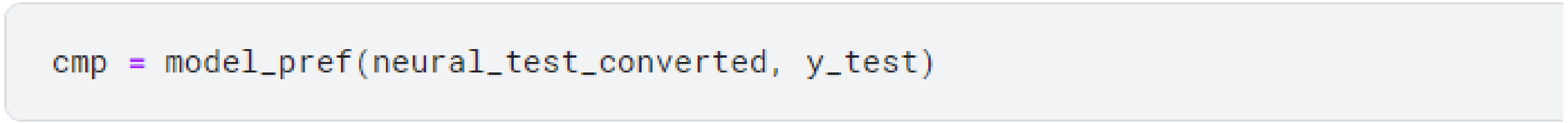


	accuracy	loss	val_accuracy	val_loss	epoch
545	0.945652	0.194108	0.83871	0.745599	545
546	0.949275	0.183601	0.83871	0.758848	546
547	0.949275	0.184281	0.83871	0.713184	547
548	0.943841	0.182873	0.83871	0.685806	548
549	0.949275	0.179686	0.83871	0.736959	549

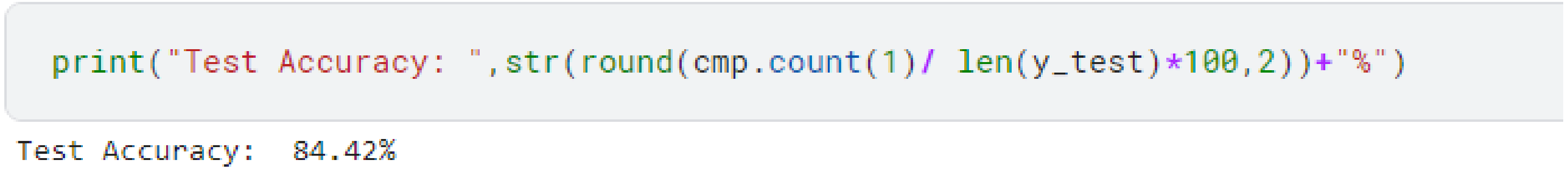
Test Accuracy



```
neural_test_converted=[]
for i in neural_test:
    if i>0.5:
        neural_test_converted.append(1)
    else:
        neural_test_converted.append(0)
```



```
cmp = model_pref(neural_test_converted, y_test)
```



```
print("Test Accuracy: ",str(round(cmp.count(1)/ len(y_test)*100,2))+"%")
```

```
Test Accuracy:  84.42%
```

Conclusion

Diabetes can be a reason for reducing life expectancy and quality. Predicting this chronic disorder earlier can reduce the risk and complications of many diseases in the long run. In this paper, an automatic diabetes prediction system using various machine learning approaches has been proposed. The open-source Pima Indian and a private dataset of female Bangladeshi patients have been used in this work.

Code File Link

https://github.com/AhmedAli085/diabetes-Prediction-Machine-Learning-/blob/main/diabetes-machine-larning-pieas.ipynb

- Thank You