AI Based Diabetes prediction system

Phase 5: Project Documentation & Submission

Problem Statement:

The primary challenge is to develop a robust AI-based system that accurately predicts diabetes risk based on patient data, including age, gender, family history, BMI, blood pressure, and glucose levels. The goal is to aid healthcare professionals in timely intervention and provide patients with preventive measures.

Design thinking process:

Data Collection:

A comprehensive dataset of patient records, including both diabetic and non-diabetic cases, was collected for model training and testing.

Machine Learning Models:

The system employs various supervised learning algorithms, including logistic regression, decision trees, random forests, and neural networks, to build predictive models.

User Interface:

A user-friendly interface for healthcare professionals allows input of patient data and displays the prediction results.

Phases of development:

This report presents the development and evaluation of an AI-based Diabetes Prediction System, aimed at early detection and management of diabetes. The system employs machine learning algorithms to predict the likelihood of an individual developing diabetes. The report provides insights into the system's objectives, design, implementation, testing, challenges, future enhancements, and concludes with recommendations for further development**.**

Dataset used:

Dataset link: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Data preprocessing:

The system was implemented using Python and popular machine learning libraries, including Scikit-Learn and TensorFlow. Data preprocessing, model training, and user interface development were integral parts of the implementation process.

Feature:

Future enhancements for the system include:

- Continuous data collection to adapt to changing health trends.
- Integration with electronic health records (EHR) systems.
- Implementation of explainable AI techniques for model interpretability.
- Expansion to include other risk factors, such as genetics and lifestyle data.

Mechine learning Algorithm:

➢ Logistic Regression

Training the Model:

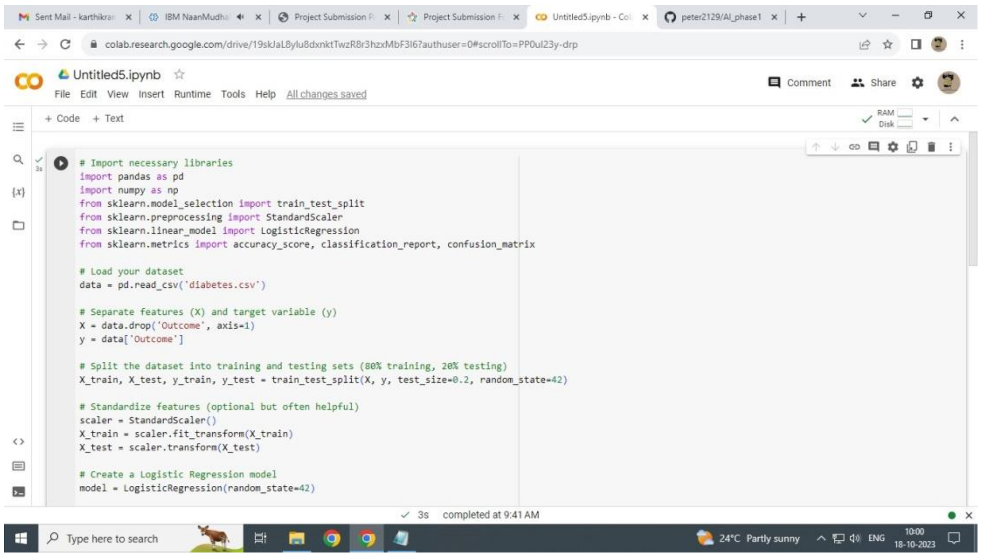
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Code:

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load your dataset
data = pd.read_csv('diabetes.csv')

# Separate features (X) and target variable (y)
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize features (optional but often helpful)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create a Logistic Regression model
model = LogisticRegression(random_state=42)
```

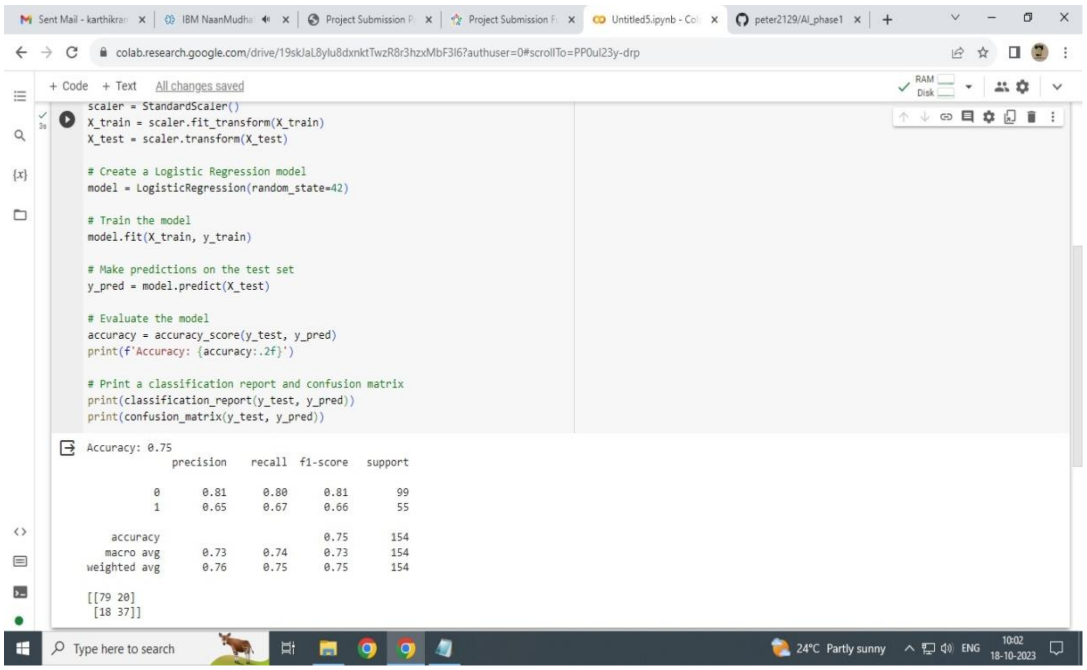
```python
# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Print a classification report and confusion matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

<u>Sample output:</u>



```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load your dataset
data = pd.read_csv('diabetes.csv')

# Separate features (X) and target variable (y)
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features (optional but often helpful)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create a Logistic Regression model
model = LogisticRegression(random_state=42)
```

✓ 3s completed at 9:41 AM

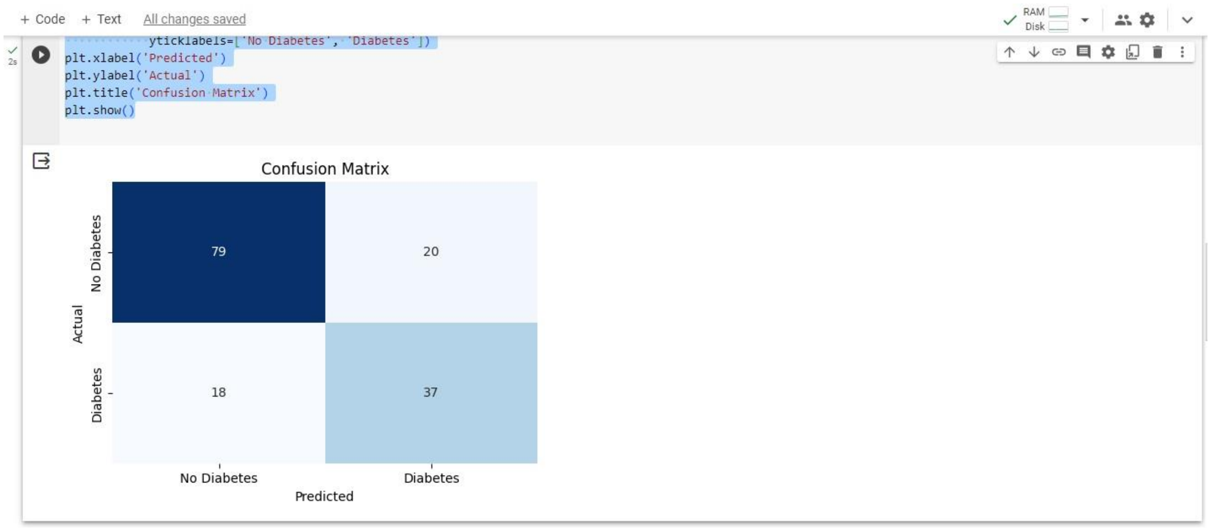


```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create a Logistic Regression model
model = LogisticRegression(random_state=42)

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

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Print a classification report and confusion matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.75
              precision    recall  f1-score   support

           0       0.81      0.80      0.81        99
           1       0.65      0.67      0.66        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154

[[79 20]
 [18 37]]
```

Visualization:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Create a heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Diabetes', 'Diabetes'],
            yticklabels=['No Diabetes', 'Diabetes'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



```
            yticklabels=['No Diabetes', 'Diabetes'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```