AI-BASED DIABETES PREDICTION SYSTEM

AI_PHASE 3

OCTOBER 18, 2023

MD AJAMTULLAH ZAFAR

This paper is Phase 3 of Md Ajamtullah Zafar "Credit Card Fraud Detection" project, which he presented as a third-year Computer Science And Engineering student. Phase 3 is dubbed "Development Part 1," and it consists of preparing the dataset, ingesting data, doing data transformation, and starting the initial data analysis.

1. Introduction

The introduction section provides an overview of your AI-based diabetes prediction project, setting the context for the entire program.

1.1. Background

In this subsection, you will provide background information on diabetes and the significance of using AI for its prediction. Discuss the prevalence of diabetes, its health implications, and the need for accurate predictive models.

1.2. Objective

- Define the main objective of your project, which is to develop an AI-based diabetes prediction model using Python.

- Mention the specific goals and outcomes you aim to achieve.

- Highlight the potential benefits of such a model, such as early detection and improved patient care.

1.3. Scope

- Clearly define the scope of your project, including the data sources you will use, the machine learning techniques you'll employ, and the target audience for the prediction model.

- Mention any constraints or limitations that might affect the scope of the project.

- Provide a brief overview of the technologies and tools you plan to utilize, such as Python libraries, frameworks, and data sources.

2. Data Collection and Preprocessing

This section focuses on the process of obtaining and preparing the data for use in the AI-based diabetes prediction model.

2.1. Data Sources

- Provide a detailed description of the data sources you are using, including the origin, format, and any relevant information about the dataset.

- Explain why you chose these data sources and their suitability for the project.

2.2. Data Cleaning

- Describe the steps involved in data cleaning, which may include handling missing values, outliers, and duplicate records.

  - Explain the data cleaning techniques you applied and their impact on the dataset's quality.

  - Provide before-and-after examples of data cleaning processes.

2.3. Data Exploration

- Discuss the process of exploring the dataset to gain insights into its characteristics.

  - Include visualizations, statistical summaries, and key findings about the data, such as distributions, correlations, and trends.

  - Describe any domain-specific knowledge that guided your exploration.

2.4. Feature Engineering

- Explain the process of feature engineering, which involves selecting, transforming, or creating features that are relevant for the prediction model.

  - Describe the features you selected and why they are important for diabetes prediction.

  - Discuss any feature scaling or normalization techniques applied to the data.

3. Machine Learning Models

This section delves into the core of your AI-based diabetes prediction system, explaining the steps involved in developing, training, and evaluating the machine learning models.

3.1. Model Selection

- Describe the machine learning algorithms or models you considered for your diabetes prediction task.

  - Explain the rationale behind your choice of models, considering factors like the nature of the data and the problem's requirements.

  - Provide an overview of each selected model, including its strengths and weaknesses.

3.2. Data Splitting (Training, Validation, Testing)

  - Explain the process of dividing the dataset into training, validation, and testing sets.

- Describe the proportions used for each split and why they were chosen.
- Discuss the importance of each set for model development, tuning, and evaluation.

3.3. Model Training

- Detail the steps involved in training the selected machine learning models.
- Include code examples and describe the parameters and techniques used for training.
- Explain how you handle overfitting and underfitting during the training process.

3.4. Hyperparameter Tuning

- Discuss the process of hyperparameter tuning to optimize the models' performance.
- Explain the methods employed, such as grid search or random search.
- Provide insights into the best hyperparameters found for each model.

3.5. Model Evaluation Metrics

- Present the evaluation metrics used to assess the performance of the trained models.
- Explain the significance of each metric, including accuracy, precision, recall, F1-score, and AUC-ROC.
- Provide the evaluation results for each model, including any comparative analysis.

4. Feature Selection and Importance

This section focuses on the process of selecting and assessing the importance of features in the context of your diabetes prediction model.

4.1. Feature Importance Analysis

- Explain the significance of feature importance in machine learning and its relevance to your diabetes prediction model.
- Detail the techniques used to assess the importance of each feature in the dataset.

4.2. Feature Selection Techniques

- Describe the methods and strategies you employed for feature selection.

- Explain why feature selection is important and how it can enhance model efficiency and interpretability.

5. Model Deployment

This section outlines the process of deploying your AI-based diabetes prediction model, making it accessible for end-users.

5.1. Flask Web Application

- Describe the development and deployment of a Flask web application that hosts your prediction model.

- Explain the structure and components of the web application, including routes, views, and templates.

5.2. API Integration

- Explain the integration of the machine learning model into the Flask application through an API (Application Programming Interface).

- Detail the endpoints and routes exposed by the API for making predictions.

5.3. User Interface

- Describe the user interface (UI) of the web application, including how users interact with the prediction model.

- Include screenshots or mockups of the UI, showcasing the input fields and prediction results.

- Discuss the design choices made for the user interface, emphasizing user-friendliness and ease of use.

6. Model Testing and Validation

This section focuses on the testing and validation of your deployed diabetes prediction model, ensuring its accuracy and reliability.

6.1. Testing the Deployment

- Describe the process of testing the deployed model within the Flask web application or API.
- Explain the test cases, scenarios, or user interactions used to assess the model's functionality.
- Include real-world use cases and results obtained from testing.

6.2. Cross-Validation

- Explain the use of cross-validation techniques to assess the model's generalization and robustness.
- Describe the specific cross-validation methods employed (e.g., k-fold cross-validation).
- Provide cross-validation results, including any insights gained from variations in training and validation datasets.
- Discuss how cross-validation impacts the model's performance and reliability.

6.3. Performance Metrics

- Detail the performance metrics used to evaluate the deployed model's predictive accuracy.
- Include quantitative measures such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC).
- Provide the results of these metrics for both testing the deployment and cross-validation.
- Compare the model's performance against initial expectations and industry standards.

7. Results and Discussion

This section presents the outcomes of your AI-based diabetes prediction project and discusses the implications and interpretability of the results.

7.1. Model Performance

- Provide a comprehensive analysis of the model's performance based on the evaluation metrics discussed in Section 3.5.

- Present the accuracy, precision, recall, F1-score, and AUC-ROC values.
- Include visual aids like charts or graphs to highlight the model's strengths and weaknesses.
- Discuss any notable patterns or trends in the performance metrics.

7.2. Interpretability of Results

- Explain the interpretability of the model's predictions and how users can understand the reasoning behind those predictions.
- Provide examples of how interpretability techniques enhance the model's transparency.

7.3. Implications

- Discuss the practical implications of your AI-based diabetes prediction model.
- Explain how the model's accuracy and interpretability can benefit healthcare professionals and patients.
- Consider potential use cases, such as early detection, personalized treatment recommendations, and health management.
- Address any limitations or challenges that may affect the model's real-world deployment.

8. Conclusion

The conclusion section provides a comprehensive summary of your AI-based diabetes prediction project, highlighting the key findings, acknowledging limitations, and suggesting avenues for future work.

8.1. Summary of Findings

- Summarize the major findings and achievements of your project, emphasizing the positive outcomes.
- Highlight the model's performance, interpretability, and any unique contributions.
- Reiterate the significance of your project in the context of diabetes prediction and healthcare.

8.2. Limitations

- Acknowledge the limitations and challenges encountered during the project.

- Discuss any data limitations, such as data quality, quantity, or representativeness.

- Address limitations related to model performance, interpretability, or practical deployment.

8.3. Future Work

- Propose ideas for future work and improvements based on the lessons learned during this project.

The "Conclusion" section serves as a reflection on the entire project, summarizing its achievements, recognizing its limitations, and providing a roadmap for future research and development.

9. References

This section includes all the sources, materials, and references used throughout your AI-based diabetes prediction project.

9.1. Data Sources

- List and cite the data sources used in your project, including any datasets, databases, or data repositories.

- Provide details on the origin of the data, publication dates, and relevant publications or documentation.

- Include URLs or references to where the data can be accessed or obtained.

9.2. Related Research

- List and cite any research papers, articles, or studies that informed or inspired your work.

- Include relevant papers on AI-based diabetes prediction, machine learning, and healthcare applications.

- Provide full bibliographic information for each source.

9.3. Python Libraries and Frameworks

- Cite the Python libraries and frameworks used in your project for data analysis, machine learning, web development, and other purposes.

- Include the names of the libraries, their versions, and references to their official documentation or websites.

10. Appendices

This section includes supplementary information that supports the main content of your AI-based diabetes prediction project.

10.1. Code Samples

- Provide code samples and snippets relevant to your project. Include examples of data preprocessing, model training, web application development, and API integration.

- Offer explanations and comments within the code to help readers understand its purpose and functionality.

- Highlight any critical or innovative portions of your code that are central to your project.

10.2. Configuration Details

- Include detailed information about the configurations, settings, and parameters used in your project.

- Share configuration files or settings related to your machine learning models, Flask application, or any other critical components.

- Explain how readers can replicate your project with similar configurations.

10.3. Glossary

- Provide a glossary of key terms, acronyms, and technical jargon used throughout your project.

- Define and explain terms specific to machine learning, healthcare, and diabetes prediction to aid readers who may be less familiar with these concepts.

- Include cross-references to the relevant sections of your project where these terms are used.

The "Appendices" section enhances the clarity and accessibility of your project by offering code samples for implementation, configuration details for replication, and a glossary for improved understanding of technical terminology.

PROGRAM:-

```
# Naive Bayes Classification
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns;

#Importing the dataset
dataset = pd.read_csv('diabetes.csv')
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, 8]

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3,
random_state = 2)


# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Training set results
y_pred = classifier.predict(X_train)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_train, y_pred)

# Fitting Naive Bayes to the Testing set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_test, y_test)

# Predicting the Testing set results
y_pred1 = classifier.predict(X_test)
```

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, y_pred1)

#create an empty data frame that we have to predict
variety=pd.DataFrame()
variety['Pregnancies']=[6]
variety['Glucose']=[148]
variety['BloodPressure']=[72]
variety['SkinThickness']=[35]
variety['Insulin']=[0]
variety['BMI']=[33.6]
variety['DiabetesPedigreeFunction']=[0.627]
variety['Age']=[50]
print(variety)

y_pred1=classifier.predict(variety)
print("the Outcome of the Patient is:")
print(y_pred1)

#Heat map of a confusion matrix
import seaborn as sns
sns.heatmap(cm,fmt=".0f",xticklabels=['Diabeties_yes','Diabeties_no'],yticklabels
=['Diabeties_yes','Diabeties_no'],annot=True)
#sns.heatmap(cm,fmt=".0f",annot=True)

#Calculating Performance Metrics for Training Set
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)
FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)
# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
print("Recall",TPR)
# Specificity or true negative rate
TNR = TN/(TN+FP)
print("Specificity",TNR)
# Precision or positive predictive value
PPV = TP/(TP+FP)
print("Precision",PPV)
```

```python
# Negative predictive value
NPV = TN/(TN+FN)
print("Negative Predictive Value",NPV)
# Fall out or false positive rate
FPR = FP/(FP+TN)
print("False Positive Rate",FPR)
# False negative rate
FNR = FN/(TP+FN)
print("False Negative Rate",FNR)
# False discovery rate
FDR = FP/(TP+FP)
print("False Discovery Rate",FDR)
# Overall accuracy for each class
ACC = (TP+TN)/(TP+FP+FN+TN)
print("Accuracry",ACC)

#Decision Boundary for Diabetes Dataset
X1 = dataset.iloc[:,[1,2]]
y1 = dataset.iloc[:, 8]

#Encoding the Dependent Variable Y
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_y = LabelEncoder()
y1 = labelencoder_y.fit_transform(y1)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size = 1/3,
random_state = 0)


# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X1_train = sc.fit_transform(X1_train)
X1_test = sc.transform(X1_test)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier2 = GaussianNB()
classifier2.fit(X1_train, y1_train)

# Predicting the Training set results
y_predimg = classifier2.predict(X1_train)
```

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cmimg = confusion_matrix(y1_train, y_predimg)

#Visulaizing the Results
a=np.arange(start=X1_train[:,0].min()-1,stop=X1_train[:,0].max()+1,step=0.01)
b=np.arange(start=X1_train[:,1].min()-1,stop=X1_train[:,1].max()+1,step=0.01)
XX,YY=np.meshgrid(a,b)

#Classifying every point on the meshgrid
input_array=np.array([XX.ravel(),YY.ravel()]).T
labels=classifier2.predict(input_array)

#Plotting an array as image using Countour Function
# Plot also the training points
plt.title("Naive Bayes Decision Boundary of Diabeties Dataset")
plt.contourf(XX,YY,labels.reshape(XX.shape),alpha=0.25)
plt.scatter(X1_train[:,0],X1_train[:,1],c=y_predimg)
plt.xlabel('Glucose')
plt.ylabel('Blood Pressure')
```