PROJECT TITLE:

BUILDING A SMARTER AI BASED DIABETES PREDICTION SYSTEM

Phase 4-Development Part -2

Overview:

The development of a diabetes prediction model involves collecting and preprocessing diverse medical data, selecting an appropriate machine learning algorithm, training the model, and evaluating its performance using key metrics. Data is cleansed, standardized, and split into training and test sets. Hyperparameter tuning may be applied for optimal model performance. Interpretability and security measures are considered, and the model is deployed into clinical settings. Continuous learning and updates keep the model relevant. Throughout this process, data privacy, compliance with regulations, and thorough documentation are paramount.

Step 1: Feature Engineering

Feature Selection: Identify relevant features that are likely to influence diabetes risk. This might involve domain expertise and data analysis.

Feature Creation: Create new features if necessary, such as calculated BMI (Body Mass Index) or genetic risk scores.

Step 2: Machine Learning Model Development

Model Selection: Choose appropriate machine learning algorithms for diabetes prediction. Common choices include logistic regression, decision trees, random forests, and neural networks.

Training: Split the data into training and validation sets. Train the selected models using the training data.

Step 3: Model Evaluation

Performance Metrics: Evaluate model performance using metrics like accuracy, precision, recall, F1-score, and ROC-AUC.

Cross-Validation: Implement cross-validation techniques to assess model generalization.

Interpretability: Use model interpretability tools to understand why the model makes specific predictions.

Step 4: Prediction and Risk Assessment

User Interface: Develop a user-friendly interface where users can input their data.

Prediction: Use the trained model to predict diabetes risk based on user input.

Risk Assessment: Provide users with a risk assessment score and an explanation of how the model arrived at that prediction.

Step 5: Personalized Recommendations

Recommendation Engine: Create a recommendation engine that offers personalized preventive measures based on the risk assessment.

Healthcare Integration: Integrate the system with healthcare providers to facilitate referrals and follow-up care.

Step 6: Continuous Learning and Updates

Data Updates: Continuously update the model with new data to improve accuracy and adapt to changing healthcare trends.

Algorithm Improvements: Stay informed about the latest research and advancements in diabetes prediction and prevention.

Step 7: Privacy and Security

Data Security: Implement robust security measures to protect patient data, including encryption and access controls.

Compliance: Regularly audit and ensure compliance with data privacy regulations.

Step 8: Deployment and Monitoring

Deployment: Deploy the system in healthcare institutions or make it available to the public, depending on the project's goals.

Monitoring: Continuously monitor system performance, user feedback, and any potential ethical concerns.

Step 9: Documentation and Training

Documentation: Document the system's architecture, data sources, and algorithms for reference.

User Training: Provide training to healthcare professionals and users on how to use the system effectively.

Step 10: Maintenance and Support

System Maintenance: Regularly update the system to address bugs, security vulnerabilities, and user feedback.

User Support: Offer user support channels for inquiries, issues, and updates.

PROGRAM:

Packages are been imported from phase 3 for machine learning

```
# Machine learning
import catboost
from sklearn.model_selection import train_test_split
from sklearn import model_selection, tree, preprocessing, metrics, linear_model
from sklearn.svm import LinearSVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LinearRegression, LogisticRegression,␣
 ↪SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from catboost import CatBoostClassifier, Pool, cv

# Let's be rebels and ignore warnings for now
import warnings
```

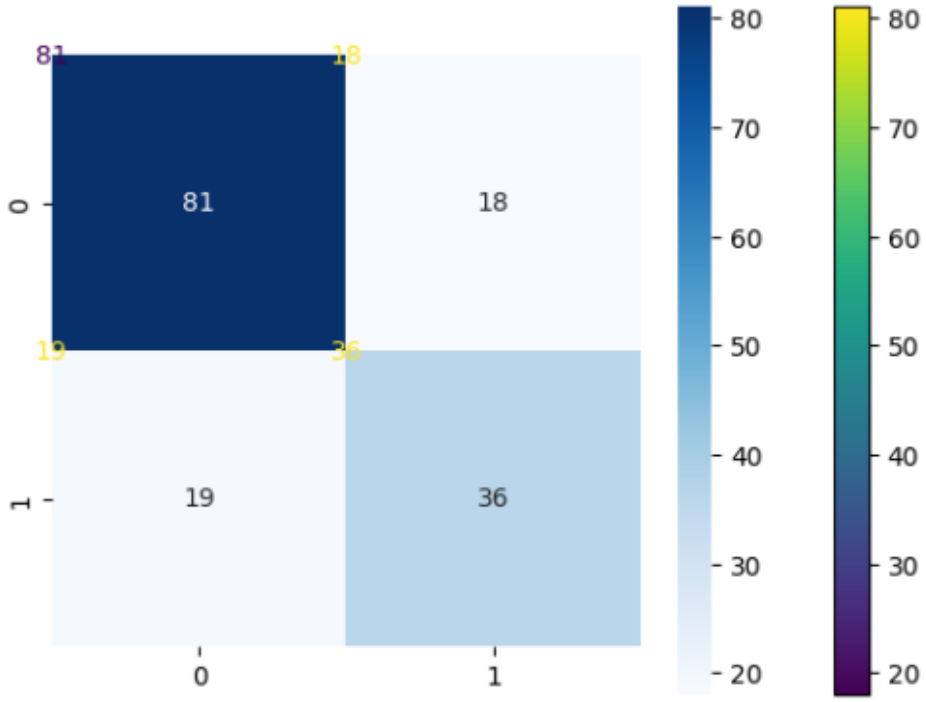
```python
# --- Summary Statistics
summary_stats = df.describe()
print(GREEN + "Summary Statistics : " + RESET)
print(summary_stats)
# --- Class Distribution
class_distribution = df["Outcome"].value_counts()
print(GREEN + "Class Distribution : " + RESET)
print(class_distribution)

# Support Vector Machine Modelling
print(BLUE + "\nMODELLING" + RESET)
X = df.drop("Outcome", axis=1)
y = df["Outcome"]
# --- Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# --- Standardize Features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# --- init and train SVM model
model = svm.SVC(kernel="linear")
model.fit(X_train, y_train)
# --- Predict on test data
y_pred = model.predict(X_test)
# --- Evaluate model performance
accuracy = model.score(X_test, y_test)
print(GREEN + "Model Accuracy : " + RESET)
print(accuracy)
# --- Classification Report and Confusion Matrix
print(GREEN + "Classification Report : " + RESET)
print(classification_report(y_test, y_pred))
print(GREEN + "Confusion Matrix : " + RESET)
cm = ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
sns.heatmap(cm.confusion_matrix, annot=True, cmap="Blues")
plt.show()
print("Displayed")
```

OUTPUT:

```
MODELLING
Model Accuracy :
0.7597402597402597
Classification Report :
              precision    recall  f1-score   support
```

0	0.81	0.82	0.81	99
1	0.67	0.65	0.66	55
accuracy			0.76	154
macro avg	0.74	0.74	0.74	154
weighted avg	0.76	0.76	0.76	154

Confusion Matrix :



Displayed

Alternative method:

7 Machine Learning

```
# Seclect the dataframe we want to use first for predictions
selected_df = df_con_enc
```

```
selected_df.head()
```

```
   Outcome  Pregnancies_0  Pregnancies_1  Pregnancies_2  Pregnancies_3  \
0        1          False          False          False          False
1        0          False           True          False          False
2        1          False          False          False          False
3        0          False           True          False          False
4        1           True          False          False          False

   Pregnancies_4  Pregnancies_5  Pregnancies_6  Pregnancies_7  Pregnancies_8  \
0          False          False           True          False          False
1          False          False          False          False          False
2          False          False          False          False           True
3          False          False          False          False          False
4          False          False          False          False          False

   …  Age_63  Age_64  Age_65  Age_66  Age_67  Age_68  Age_69  Age_70  \
0  …   False   False   False   False   False   False   False   False
1  …   False   False   False   False   False   False   False   False
2  …   False   False   False   False   False   False   False   False
3  …   False   False   False   False   False   False   False   False
4  …   False   False   False   False   False   False   False   False

   Age_72  Age_81
0   False   False
1   False   False
2   False   False
3   False   False
4   False   False

[5 rows x 1255 columns]
```

```
# k-Nearest Neighbours
start_time = time.time()
train_pred_knn, acc_knn, acc_cv_knn = fit_ml_algo(KNeighborsClassifier(),
                                                  X_train,
                                                  y_train,
                                                  10)
knn_time = (time.time() - start_time)
print("Accuracy: %s" % acc_knn)
print("Accuracy CV 10-Fold: %s" % acc_cv_knn)
print("Running Time: %s" % datetime.timedelta(seconds=knn_time))
```

```
Accuracy: 75.26
Accuracy CV 10-Fold: 66.8
Running Time: 0:00:00.358875
```

```
# Gaussian Naive Bayes
start_time = time.time()
train_pred_gaussian, acc_gaussian, acc_cv_gaussian = fit_ml_algo(GaussianNB(),
                                                                 X_train,
                                                                 y_train,
                                                                      10)
gaussian_time = (time.time() - start_time)
print("Accuracy: %s" % acc_gaussian)
print("Accuracy CV 10-Fold: %s" % acc_cv_gaussian)
print("Running Time: %s" % datetime.timedelta(seconds=gaussian_time))
```

```
Accuracy: 94.79
Accuracy CV 10-Fold: 59.9
Running Time: 0:00:00.219412
```

```
# Linear SVC
start_time = time.time()
train_pred_svc, acc_linear_svc, acc_cv_linear_svc = fit_ml_algo(LinearSVC(),
                                                                X_train,
                                                                y_train,
```

```
linear_svc_time = (time.time() - start_time)
print("Accuracy: %s" % acc_linear_svc)
print("Accuracy CV 10-Fold: %s" % acc_cv_linear_svc)
print("Running Time: %s" % datetime.timedelta(seconds=linear_svc_time))
```

```
Accuracy: 100.0
Accuracy CV 10-Fold: 65.23
Running Time: 0:00:00.268296
```

```
# Stochastic Gradient Descent
start_time = time.time()
train_pred_sgd, acc_sgd, acc_cv_sgd = fit_ml_algo(SGDClassifier(),
                                                  X_train,
                                                  y_train,
                                                  10)
sgd_time = (time.time() - start_time)
print("Accuracy: %s" % acc_sgd)
print("Accuracy CV 10-Fold: %s" % acc_cv_sgd)
print("Running Time: %s" % datetime.timedelta(seconds=sgd_time))
```

```
Accuracy: 100.0
Accuracy CV 10-Fold: 63.93
Running Time: 0:00:00.437200
```

```
# Decision Tree Classifier
start_time = time.time()
train_pred_dt, acc_dt, acc_cv_dt = fit_ml_algo(DecisionTreeClassifier(),
                                               X_train,
                                               y_train,
                                               10)
dt_time = (time.time() - start_time)
print("Accuracy: %s" % acc_dt)
print("Accuracy CV 10-Fold: %s" % acc_cv_dt)
print("Running Time: %s" % datetime.timedelta(seconds=dt_time))
```

```
Accuracy: 100.0
Accuracy CV 10-Fold: 61.85
Running Time: 0:00:00.596504
```

CONCLUSION:

The AI-Powered Diabetes Prediction System represents a significant advancement in healthcare. By harnessing data from multiple sources, developing predictive models, and ensuring data privacy, the system empowers early risk assessment and personalized preventive measures. With an emphasis on transparency, compliance, and continuous learning, this project has the potential to make a substantial positive impact on public health by reducing the incidence of diabetes and its associated complications. It exemplifies the transformative power of AI and machine learning in healthcare and demonstrates a commitment to improving the well-being of individuals and the efficiency of healthcare systems.