AI BASED DIABETES PREDICTION SYSTEYM

Development Phase 1: Data Preparation and Model Training

1. *Data Collection and Integration*:

In this phase, you collect and integrate data from various sources, such as electronic health records and genetic information. You'll typically load the data into a format suitable for analysis, like a pandas DataFrame in Python.

```
python
import pandas as pd

# Load patient data from a CSV file
patient_data = pd.read_csv('patient_data.csv')

# Load genetic data from another source (e.g., CSV or database)
genetic_data = pd.read_csv('genetic_data.csv')
```

2. *Data Preprocessing*:

Data preprocessing involves cleaning and transforming data to make it suitable for machine learning. Common preprocessing tasks include handling missing values, encoding categorical data, and scaling numeric features.

```
python
# Handle missing values (e.g., fill with mean or median)
patient_data.fillna(patient_data.mean(), inplace=True)

# Encode categorical features (if any)
patient_data = pd.get_dummies(patient_data, columns=['gender', 'smoker'])

# Scale numeric features (e.g., to a 0-1 range)
from sklearn.preprocessing import MinMaxScaler
```

```python
scaler = MinMaxScaler()
patient_data[['age', 'bmi']] = scaler.fit_transform(patient_data[['age', 'bmi']])
```

3. *Feature Engineering*:

Feature engineering involves selecting and creating relevant features that can influence diabetes risk prediction. For example, you might use statistical tests to select the most important features.

python

```python
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

# Define the target variable (diabetes status)
X = patient_data.drop('diabetes_status', axis=1)
y = patient_data['diabetes_status']

# Use ANOVA F-statistic to select the top k features
selector = SelectKBest(score_func=f_classif, k='all')
X_new = selector.fit_transform(X, y)
```

4. *Model Selection*:

Choose a machine learning algorithm for diabetes prediction. In this example, we'll use a Random Forest classifier from scikit-learn.

python

```python
from sklearn.ensemble import RandomForestClassifier

# Initialize the Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
```

5. *Model Training*:

Split the data into training and testing sets and train the selected model on the training data.

```python
from sklearn.model_selection import train_test_split

# Split data into training and testing sets (e.g., 80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2, random_state=42)

# Train the Random Forest classifier
clf.fit(X_train, y_train)
```

6. *Model Evaluation*:

Assess the model's performance using evaluation metrics like accuracy, precision, recall, and F1-score on the test data.

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
```

```
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```