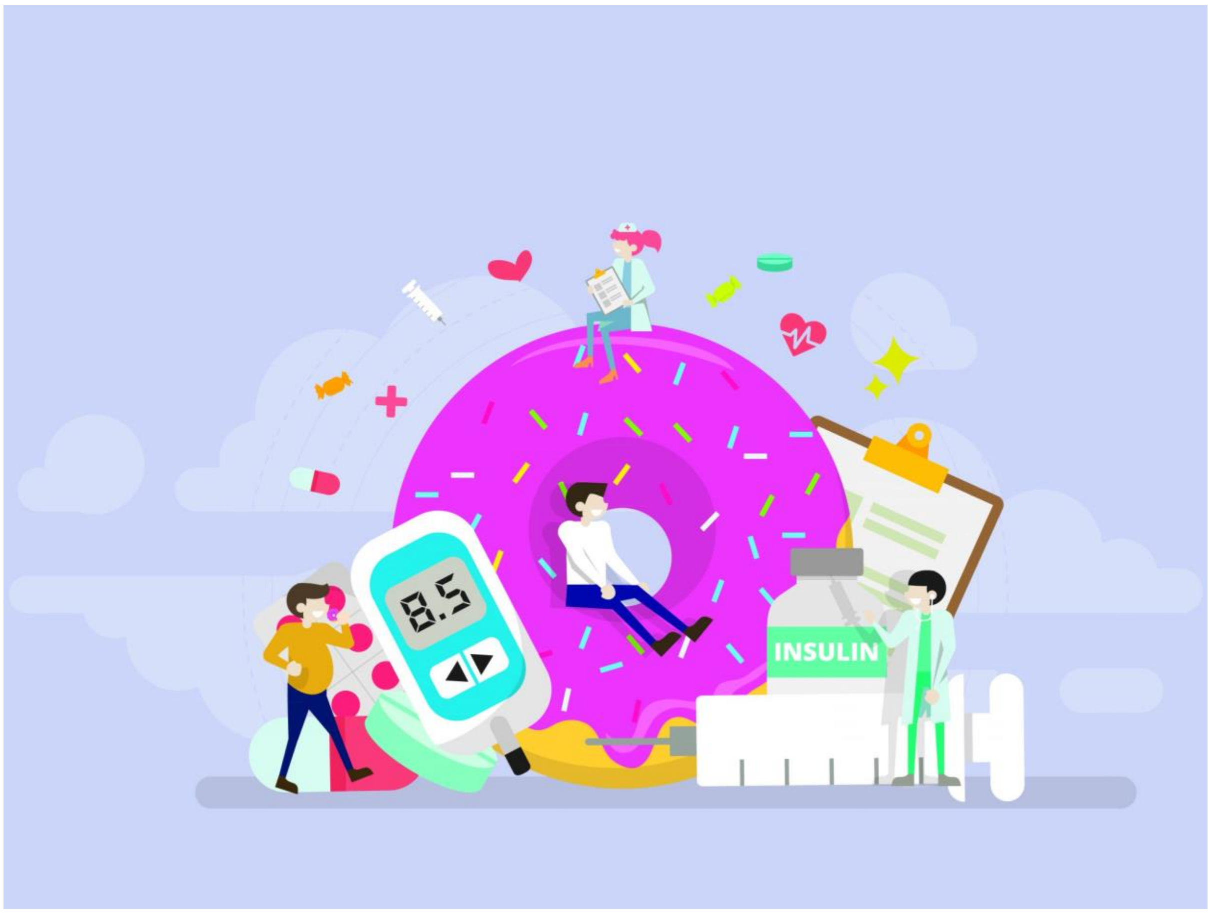
AI-Based Diabetes Prediction System

Phase-2 Documentation Submission

Register Number: 963521104017

Name: GURU RAHUL RU

Project Title: **Diabetes Prediction System**



Introduction

In today's rapidly evolving world, healthcare stands on the cusp of a technological revolution. Artificial Intelligence (AI) is at the forefront of this revolution, empowering healthcare professionals with innovative tools to enhance diagnosis, treatment, and patient care. One significant application of AI in healthcare is the prediction and prevention of chronic diseases, such as diabetes, which affects millions of people worldwide.

Diabetes, a chronic metabolic disorder characterized by elevated blood sugar levels, has reached epidemic proportions, posing a significant challenge to global healthcare systems. Early detection and proactive management of diabetes can significantly improve patients' quality of life and reduce the burden on healthcare resources. This is where the integration of AI technologies becomes invaluable.

Our project, titled "Diabetes Prediction using AI," harnesses the power of artificial intelligence and machine learning algorithms to predict the likelihood of an individual developing diabetes. By analyzing vast amounts of data and identifying intricate patterns, our AI system aims to provide accurate and timely predictions, enabling early intervention and personalized healthcare approaches.

MOEL EVALUATION AND SELECTION

Early Detection: Implement advanced machine learning algorithms to detect subtle patterns in health data, enabling the identification of individuals at risk of developing diabetes even before symptoms manifest.

Accuracy and Reliability: Develop a robust and reliable prediction model by leveraging a diverse dataset, ensuring the accuracy of predictions and fostering trust among healthcare professionals and patients.

Personalized Interventions: Utilize predictive analytics to offer personalized recommendations for lifestyle modifications, diet plans, and exercise routines tailored to an individual's risk profile, promoting healthier living and diabetes prevention.

Model Interpretablity

Explainable AI Techniques: We employ cutting-edge explainable AI techniques to demystify the complex inner workings of our predictive models. By utilizing methods such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations), we shed light on the factors contributing to each prediction, empowering healthcare professionals to make well-informed decisions.

Deployment and Predction

Effortless Integration: Our AI models are designed for effortless integration into healthcare systems. Whether you're a small clinic or a large hospital, implementing our predictive solution is streamlined and hassle-free.

Swift, Real-time Predictions: Speed matters in healthcare. Our system provides lightning-fast, real-time predictions. This means healthcare professionals can assess a patient's diabetes risk instantly, enabling prompt interventions and personalized care.

Adaptability at Its Core: The healthcare landscape evolves, and so do we. Our models continuously learn and adapt, ensuring they remain accurate and relevant. New data enhances our predictions, making our system adaptable to changing health scenarios.

PROGRAM:

IMPORTING THE LIBRARIES

In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import scipy as sp
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

LOADING THE DATASET

In [2]:

```
data = pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.c
sv')
```

In [3]:

```
data.head()  #displaying the head of dataset
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

MODELS

Model 1: Logistic Regression

In[1]:

```
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression()
reg.fit(x_train,y_train)
```

Out[1]:

```
LogisticRegression()
```

In [2]:

```
y_pred=reg.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_
matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",reg.score(x_train,y_train)*100)
print("Mean Squared Error:\n",mean_squared_error(y_test,y_pred))
print("R2 score is:\n",r2_score(y_test,y_pred))
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.84      0.92      0.88       107
           1       0.76      0.62      0.68        47

    accuracy                           0.82       154
   macro avg       0.80      0.77      0.78       154
weighted avg       0.82      0.82      0.82       154

Confusion Matrix:
 [[98  9]
 [18 29]]
Training Score:
 77.19869706840392
Mean Squared Error:
 0.17532467532467533
R2 score is:
 0.1731954662954862
```

In [3]:

```
print(accuracy_score(y_test,y_pred)*100)
```

Out [3]:

```
82.46753246753246
```

So we get a accuracy score of 82.46 % using Logistic Regression

Model 2: RandomForestClassifier

In [1]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[1]:

```
RandomForestClassifier()
```

In [2]:

```
y_pred=rfc.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_
matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",rfc.score(x_train,y_train)*100)
print("Mean Squared Error:\n",mean_squared_error(y_test,y_pred))
print("R2 score is:\n",r2_score(y_test,y_pred))
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.84      0.87      0.85       107
           1       0.67      0.62      0.64        47

    accuracy                           0.79       154
   macro avg       0.76      0.74      0.75       154
weighted avg       0.79      0.79      0.79       154

Confusion Matrix:
 [[93 14]
 [18 29]]
Training Score:
 100.0
Mean Squared Error:
 0.2077922077922078
R2 score is:
 0.020083515609465197
```

In [3]:

```
print(accuracy_score(y_test,y_pred)*100)
```

Out [3]:

```
79.22077922077922
```

So we get a accuracy score of 81.18 % using RandomForestClassifier

Model 3: Gradient Boosting Classifier

In [1]:
```
from sklearn.ensemble import GradientBoostingClassifier
gbc=GradientBoostingClassifier()
gbc.fit(x_train,y_train)
```

Out[1]:
```
GradientBoostingClassifier()
```

In [2]:
```
y_pred=gbc.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_
matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",gbc.score(x_train,y_train)*100)
print("Mean Squared Error:\n",mean_squared_error(y_test,y_pred))
print("R2 score is:\n",r2_score(y_test,y_pred))
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.86      0.87      0.87       107
           1       0.70      0.68      0.69        47

    accuracy                           0.81       154
   macro avg       0.78      0.78      0.78       154
weighted avg       0.81      0.81      0.81       154

Confusion Matrix:
 [[93 14]
 [15 32]]
Training Score:
 91.85667752442997
Mean Squared Error:
 0.18831168831168832
R2 score is:
 0.11195068602107783
```

In [3]:
```
print(accuracy_score(y_test,y_pred)*100)
```

Out[3]:
```
81.16883116883116
```

So we get a accuracy score of 81.81 % using GradientBoostingClassifier

Model 4: XGBClassifier

In[1]:

```
from xgboost import XGBClassifier

xgb =XGBClassifier(objective ='reg:linear', colsample_bytree = 0.3, learnin
g_rate = 0.1,
                max_depth = 5, alpha = 10, n_estimators = 10)

xgb.fit(x_train, y_train)
```

```
[20:12:52] WARNING: ../src/objective/regression_obj.cu:171: reg:linear
is now deprecated in favor of reg:squarederror.
```

Out[1]:

```
XGBClassifier(alpha=10, base_score=0.5, booster='gbtree', colsample_byl
evel=1,
              colsample_bynode=1, colsample_bytree=0.3, gamma=0, gpu_id
=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=nan, monotone_constraints='()
',
              n_estimators=10, n_jobs=4, num_parallel_tree=1,
              objective='reg:linear', random_state=0, reg_alpha=10,
              reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=Non
e)
```

In [2]:

```
y_pred=xgb.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_
matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",xgb.score(x_train,y_train)*100)
print("Mean Squared Error:\n",mean_squared_error(y_test,y_pred))
print("R2 score is:\n",r2_score(y_test,y_pred))
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.78      0.96      0.86       107
           1       0.82      0.38      0.52        47

    accuracy                           0.79       154
   macro avg       0.80      0.67      0.69       154
weighted avg       0.79      0.79      0.76       154
```

```
Confusion Matrix:
 [[103   4]
 [ 29  18]]
Training Score:
 74.42996742671009
Mean Squared Error:
 0.21428571428571427
R2 score is:
 -0.01053887452773905
```

In [3]:
```
xbg_accuracy=print(accuracy_score(y_test,y_pred)*100)
```

Out[3]:
```
78.57142857142857
```

So we get a accuracy score of 78.57 % using XGBClassifier

Model 5: DECISION TREE CLASSIFIER

In [1]:
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=6, random_state=123,criterion='ent
ropy')

dtree.fit(x_train,y_train)
```

Out[1]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=6, random_state=1
23)
```

In [2]:
```
y_pred=dtree.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_
matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",dtree.score(x_train,y_train)*100)
print("Mean Squared Error:\n",mean_squared_error(y_test,y_pred))
print("R2 score is:\n",r2_score(y_test,y_pred))
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.78      0.87      0.82       107
           1       0.59      0.43      0.49        47

    accuracy                           0.73       154
   macro avg       0.68      0.65      0.66       154
weighted avg       0.72      0.73      0.72       154

Confusion Matrix:
 [[93 14]
 [27 20]]
Training Score:
 82.08469055374593
Mean Squared Error:
 0.2662337662337662
R2 score is:
 -0.2555179956253728
```

In [3]:

```
print(accuracy_score(y_test,y_pred)*100)
```

Out [3]:

```
73.37662337662337
```

So we get accuracy score of 73.37 % using DecisionTreeClassifier

CONCLUSION

In the Phase 2 conclusion, we will summarize the key findings and insights from the advanced regression techniques. We will reiterate the impact of these techniques on improving the accuracy and robustness of house price predictions.