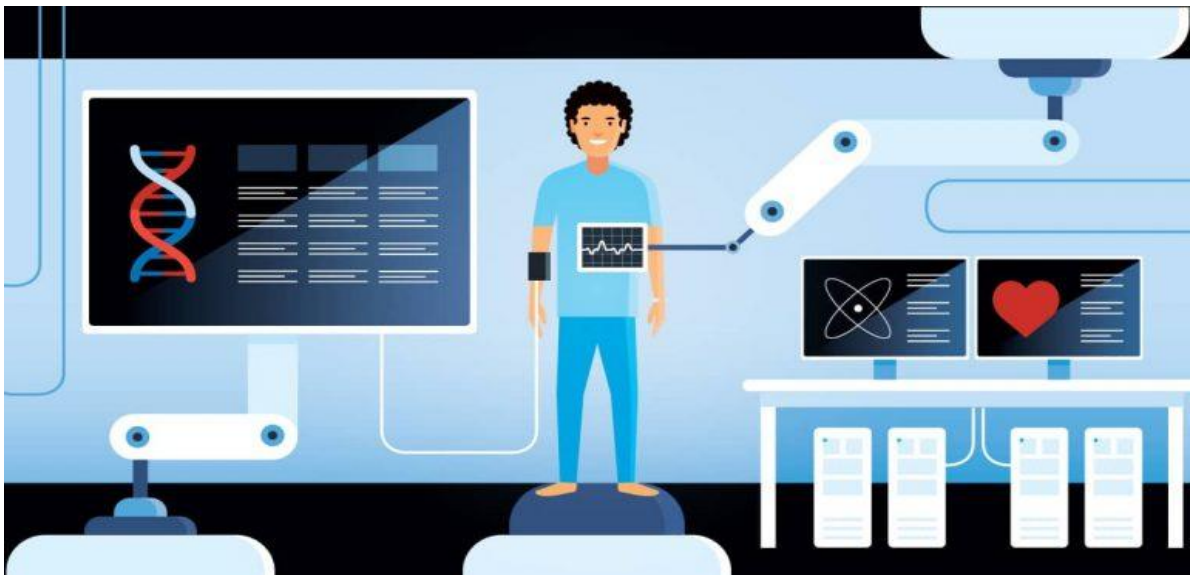
AI Based Diabetes Prediction System

TEAM MEMBER

311521106081-SAI HARSHAVARADHAN

PHASE 2 DOCUMENT SUBMISSION

INTRODUCTION



An AI-based diabetes prediction system is a computer program that uses artificial intelligence to predict whether a person is likely to develop diabetes. These systems are trained on large datasets of medical records, including information about patients' demographics, medical history, and laboratory results. The system learns to identify patterns in the data that are associated with diabetes, and then uses these patterns to predict whether a new patient is at risk.

AI-based diabetes prediction systems can be used to improve early detection and prevention of diabetes. By identifying people who are at high risk for developing diabetes, healthcare providers can recommend lifestyle changes and other interventions to help them reduce their risk. This can help to delay or prevent the onset of diabetes and its complications.

AI-based diabetes prediction systems are still under development, but they have the potential to revolutionize the way that diabetes is diagnosed and managed. By providing early warning signs of diabetes, these systems can help people to take steps to prevent or delay the onset of the disease.

DATA SOURCE:

A good data source containing medical features such as glucose levels, blood pressure, BMI, etc., along with information about whether the individual has diabetes or not.

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

Pregnancie	Glucose	BloodPres	SkinThickn	Insulin	BMI	DiabetesP	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1

Feature selection

This module involves identifying the most relevant features from the preprocessed data that are predictive of diabetes. This can be done using statistical methods or machine learning algorithms.

Model training

This module involves training a machine learning model to predict diabetes based on the selected features. There are many different machine learning algorithms that can be used for this task, such as logistic regression, decision trees, random forests, and support vector machines.

Model evaluation

This module involves evaluating the performance of the trained model on a held-out test set. This is done to assess the accuracy of the model and to identify any areas where it needs to be improved.

Model deployment:

Once the model has been trained and evaluated, it can be deployed to production so that it can be used to predict diabetes in new patients. This can be done by developing a web application, mobile app, or integrating the model with an existing electronic health record system.

PROGRAM

Ensemble methods

1:Random Forest

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision
_score, recall_score
# Load the dataset
data = pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")
data.head()
# Perform exploratory data analysis
# Summary statistics
summary_stats = data.describe()
summary_stats
# Class distribution
class_distribution = data['Outcome'].value_counts()
class_distribution
```

OUTPUT

	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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
Outcome
0    500
1    268
Name: count, dtype: int64
```

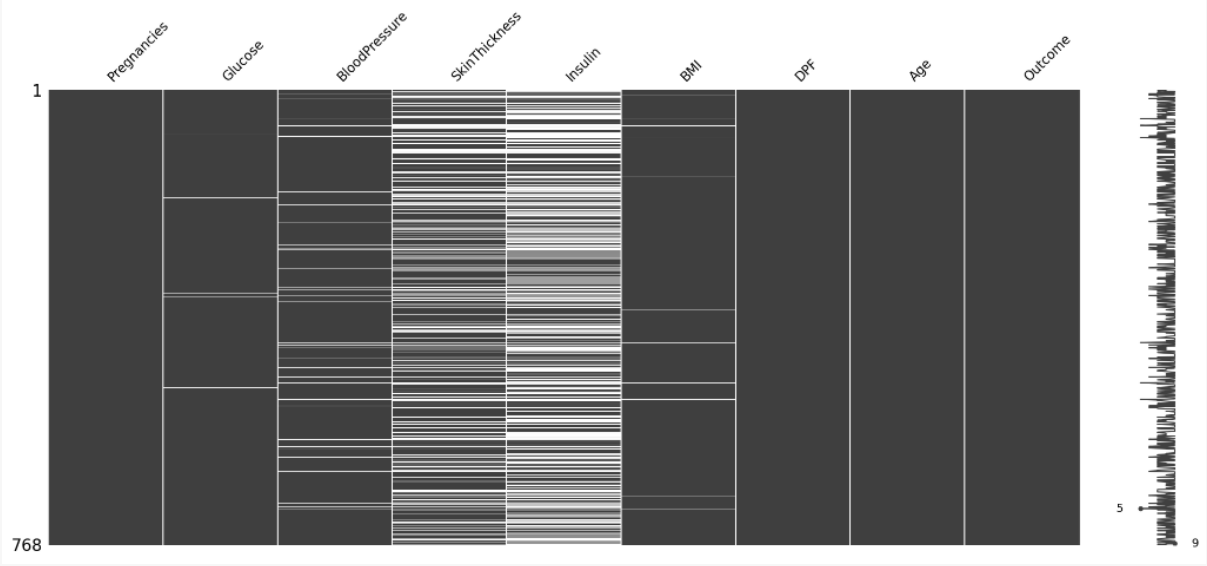
2:GradientBoosting

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
# !pip install missingno
import missingno as msno
from datetime import date
from sklearn.model_selection import train_test_split
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardS
caler, RobustScaler
# Data processing, metrics and modeling
from sklearn.metrics import f1_score, precision_score, recall_score, co
nfusion_matrix, roc_curve, precision_recall_curve, accuracy_score, roc_
auc_score
from imblearn.over_sampling import SMOTE
# Machine Learning Libraries
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingCl
assifier
from sklearn.tree import DecisionTreeClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
pd.set_option('display.width', 500)
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
def load():
    data = pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")
    return data
df = load()
df.rename(columns ={"DiabetesPedigreeFunction":"DPF"},inplace=True)
df.describe([0.01, 0.05, 0.75, 0.90, 0.99]).T
msno.matrix(df)
plt.show()
```

OUTPUT

	count	mean	std	min	1%	5%	50%	75%	90%	99%	max
Pregnancies	768.000	3.845	3.370	0.000	0.000	0.000	3.000	6.000	9.000	13.000	17.000
Glucose	768.000	120.895	31.973	0.000	57.000	79.000	117.000	140.250	167.000	196.000	199.000
BloodPressure	768.000	69.105	19.356	0.000	0.000	38.700	72.000	80.000	88.000	106.000	122.000
SkinThickness	768.000	20.536	15.952	0.000	0.000	0.000	23.000	32.000	40.000	51.330	99.000
Insulin	768.000	79.799	115.244	0.000	0.000	0.000	30.500	127.250	210.000	519.900	846.000
BMI	768.000	31.993	7.884	0.000	0.000	21.800	32.000	36.600	41.500	50.759	67.100
DPF	768.000	0.472	0.331	0.078	0.095	0.140	0.372	0.626	0.879	1.698	2.420
Age	768.000	33.241	11.760	21.000	21.000	21.000	29.000	41.000	51.000	67.000	81.000
Outcome	768.000	0.349	0.477	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000



Deep learning architectures

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) wil
list all files under the input directory
```

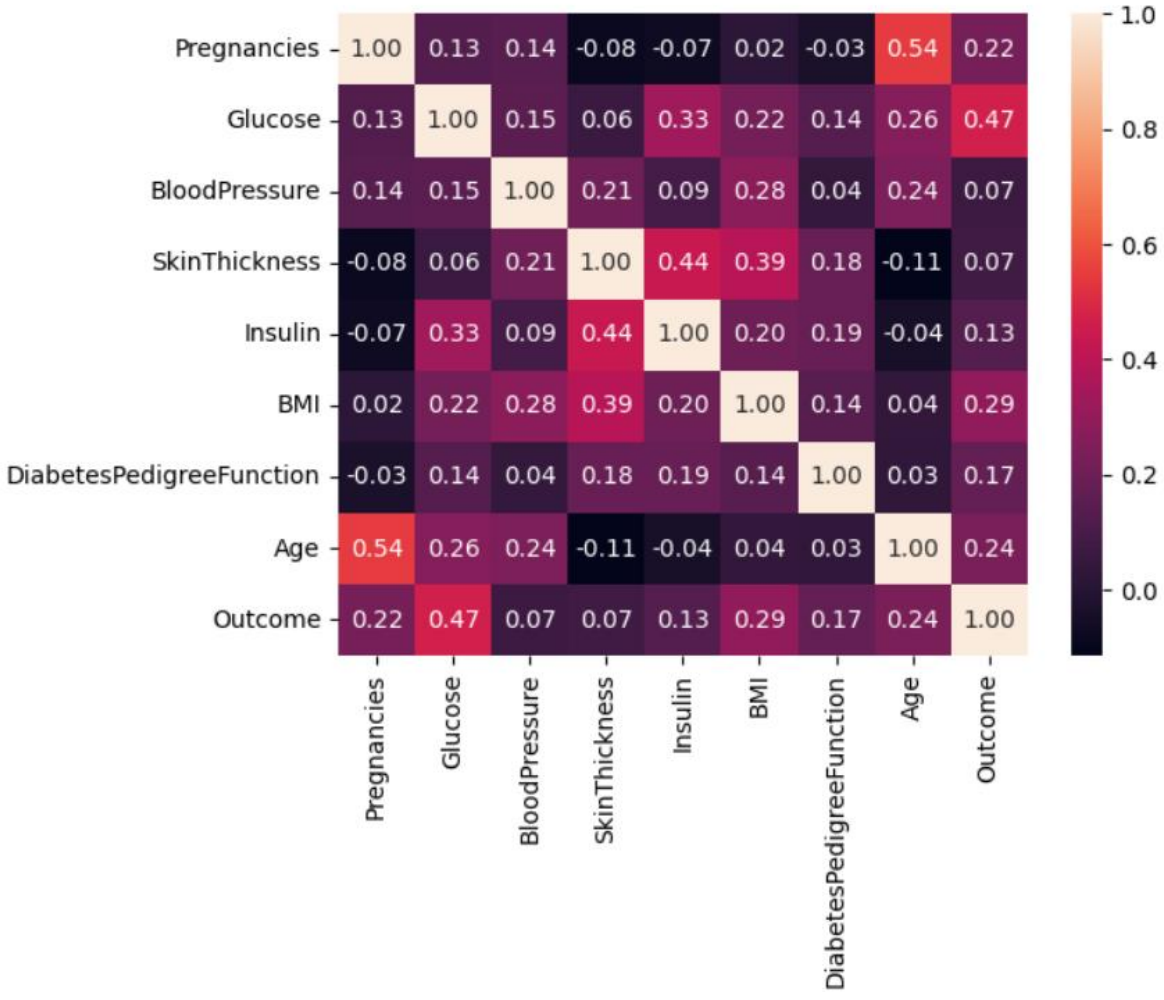
```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) th
at gets preserved as output when you create a version using "Save & Run
All"
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classific
ation_report
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from imblearn.over_sampling import RandomOverSampler
from sklearn.preprocessing import StandardScaler
data.info()
data.head(5)
data.describe()
sns.heatmap(data.corr(),annot=True,fmt='0.2f')
```

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

	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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000



CONCLUSION

AI-based systems can identify people who are at high risk for developing diabetes before they experience any symptoms. This allows for early intervention to prevent or delay the onset of the disease.Overall, AI-based diabetes prediction systems have the potential to make a significant impact on the prevention and management of diabetes.