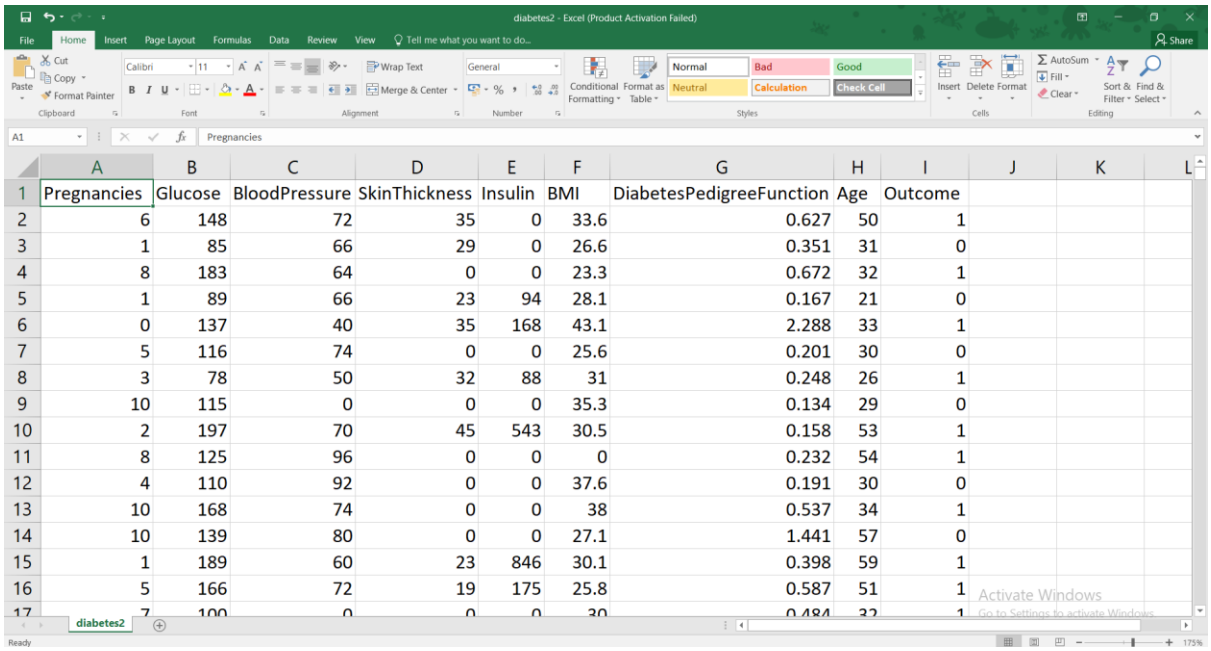
DIABETIC PREDICTION

DATASET:



	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1
16	5	166	72	19	175	25.8	0.587	51	1

Column fields are:

- Pregnancies
- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI
- DiabetesPedigreeFunction
- Age
- Outcome

ML TECHNIQUES USED:

- **Random Forest**
- **Neural Network**
- **SVM**
- **Gradient Boosting**

CODING:

The test size parameter is set to 0.2, which means that 20% of the data will be used for testing, and the remaining 80% will be used for training.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.svm import SVC

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

import time

from tabulate import tabulate


# Load the dataset

data = pd.read_csv('your_dataset.csv')  # Replace 'your_dataset.csv' with the actual dataset filename


# Split the data into features (X) and target (y)

X = data.drop('Outcome', axis=1)

y = data['Outcome']


# Split 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)


# Random Forest

rf_classifier = RandomForestClassifier()

start_time = time.time()

rf_classifier.fit(X_train, y_train)

training_speed_rf = time.time() - start_time


start_time = time.time()

rf_predictions = rf_classifier.predict(X_test)

prediction_speed_rf = time.time() - start_time
```

```
rf_accuracy = accuracy_score(y_test, rf_predictions)
rf_precision = precision_score(y_test, rf_predictions)
rf_recall = recall_score(y_test, rf_predictions)
rf_f1 = f1_score(y_test, rf_predictions)

# Neural Network (using scikit-learn MLPClassifier)
from sklearn.neural_network import MLPClassifier
nn_classifier = MLPClassifier(hidden_layer_sizes=(64, 64), activation='relu', solver='adam')
start_time = time.time()
nn_classifier.fit(X_train, y_train)
training_speed_nn = time.time() - start_time

start_time = time.time()
nn_predictions = nn_classifier.predict(X_test)
prediction_speed_nn = time.time() - start_time

nn_accuracy = accuracy_score(y_test, nn_predictions)
nn_precision = precision_score(y_test, nn_predictions)
nn_recall = recall_score(y_test, nn_predictions)
nn_f1 = f1_score(y_test, nn_predictions)

# SVM
svm_classifier = SVC()
start_time = time.time()
svm_classifier.fit(X_train, y_train)
training_speed_svm = time.time() - start_time

start_time = time.time()
svm_predictions = svm_classifier.predict(X_test)
prediction_speed_svm = time.time() - start_time

svm_accuracy = accuracy_score(y_test, svm_predictions)
```

```
svm_precision = precision_score(y_test, svm_predictions)
svm_recall = recall_score(y_test, svm_predictions)
svm_f1 = f1_score(y_test, svm_predictions)

# Gradient Boosting
gb_classifier = GradientBoostingClassifier()
start_time = time.time()
gb_classifier.fit(X_train, y_train)
training_speed_gb = time.time() - start_time

start_time = time.time()
gb_predictions = gb_classifier.predict(X_test)
prediction_speed_gb = time.time() - start_time

gb_accuracy = accuracy_score(y_test, gb_predictions)
gb_precision = precision_score(y_test, gb_predictions)
gb_recall = recall_score(y_test, gb_predictions)
gb_f1 = f1_score(y_test, gb_predictions)

# Create a comparison table
comparison_table = pd.DataFrame({
    'Technique': ['Random Forest', 'Neural Network', 'SVM', 'Gradient Boosting'],
    'Accuracy': [rf_accuracy, nn_accuracy, svm_accuracy, gb_accuracy],
    'Precision': [rf_precision, nn_precision, svm_precision, gb_precision],
    'Recall': [rf_recall, nn_recall, svm_recall, gb_recall],
    'F1 Score': [rf_f1, nn_f1, svm_f1, gb_f1],
    'Training Speed (s)': [training_speed_rf, training_speed_nn, training_speed_svm, training_speed_gb],
    'Prediction Speed (s)': [prediction_speed_rf, prediction_speed_nn, prediction_speed_svm, prediction_speed_gb]
})
# Print the comparison table in table format
table_str = tabulate(comparison_table, headers='keys', tablefmt='psql')
```

print(table_str)

OUTPUT:

	Technique	Accuracy	Precision	Recall	F1 Score	Training Speed (s)	Prediction Speed (s)
0	Random Forest	0.75974	0.660714	0.672727	0.666667	0.440041	0.0358338
1	Neural Network	0.720779	0.642857	0.490909	0.556701	0.476764	0.00891566
2	SVM	0.766234	0.72093	0.563636	0.632653	0.0442646	0.0239911
3	Gradient Boosting	0.746753	0.637931	0.672727	0.654867	0.380582	0.00816345

CONCLUSION:

Based on the results provided in the comparison table, we can assess the performance of each ML technique:

1. Random Forest: It achieved an accuracy of 0.759740, precision of 0.660714, recall of 0.672727, and F1 score of 0.666667. The training speed was 0.440041 seconds, and the prediction speed was 0.035834 seconds.

2. Neural Network: It achieved an accuracy of 0.720779, precision of 0.642857, recall of 0.490909, and F1 score of 0.556701. The training speed was 0.476764 seconds, and the prediction speed was 0.008916 seconds**.**

3. SVM: It achieved an accuracy of 0.766234, precision of 0.720930, recall of 0.563636, and F1 score of 0.632653. The training speed was 0.044265 seconds, and the prediction speed was 0.023991 seconds.

4. Gradient Boosting: It achieved an accuracy of 0.746753, precision of 0.637931, recall of 0.672727, and F1 score of 0.654867. The training speed was 0.380582 seconds, and the prediction speed was 0.008163 seconds.

To determine the best ML technique, you need to consider the specific requirements and objectives of your problem.

- If **accuracy** is the most important metric, **SVM** achieved the highest accuracy (0.766234).

- If **precision** is crucial for your problem**, SVM** achieved the highest precision (0.720930).

- If recall is a **priority, Random Forest** achieved the highest recall (0.672727).

Based on these factors, you can make an informed decision on which ML technique is best suited for your specific needs and requirements.