ST-Data Science

CSIT at UDC

Michael Travers

Diabetes Prediction

Introduction

- Diabetes is one of the most prevalent chronic diseases in the United States
- Affects millions and a financial burden on health care system
- Early diagnosis is crucial for enabling timely lifestyle modifications and medical interventions to reduce severe complications for not just diabetes but also heart disease, vision loss and kidney failure

Introduction

- This research aims to determine the most effective classification method, contributing to improved predictive analytics in healthcare.

- The models to be compared include Support Vector Machine (SVM), Decision Trees, Linear Regression, Neural Networks, which detect complex patterns through layered computation.

- This study utilizes a Kaggle dataset from 2025, comprising 100000 participants and 12 health-related features, including chronic conditions and preventive measures, to compare various machine learning models for diabetes prediction.

Objective

- Develop and compare machine learning models to determine the most accurate and efficient classification method for diabetes prediction using a large-scale health dataset.

- Analyze the impact of key health-related features on diabetes occurrence by leveraging predictive modeling techniques, aiding in early diagnosis and preventive healthcare strategies.

- Enhance public health decision-making by identifying the most effective machine learning approach for risk assessment, contributing to improved diabetes management and resource allocation.

What is diabetes

- Diabetes refers to a group of conditions characterized by a high level of blood glucose/blood sugar.which can cause serious or fatal health problems
-
- As carbohydrates break down into glucose that is carried by the blood stream to various organs in the body.
-
- beta cells in the pancreas produce insulin, insulin binds to its receptors on target cells and induces glucose intake
-
- Insulin is a hormone produced by beta cells of the pancreas and is neassary for glucose intake by the target cells
-
- Diabetes occur when the pancreas is not producing enough insulin or there is

Dataset

This dataset is from a study in 2025 that containing a total of 10000 participates

Diabetes risk factors and associated health metrics includes: Gender, Age, Hypertension, Heart disease, Smoking History, BMI, HbA1c, Glucose level, and diabetes

1 to 100 of 100000 entries Filter

gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
Female	80.0	0	1	never	25.19	6.6	140	0
Female	54.0	0	0	No Info	27.32	6.6	80	0
Male	28.0	0	0	never	27.32	5.7	158	0
Female	36.0	0	0	current	23.45	5.0	155	0
Male	76.0	1	1	current	20.14	4.8	155	0
Female	20.0	0	0	never	27.32	6.6	85	0
Female	44.0	0	0	never	19.31	6.5	200	1
Female	79.0	0	0	No Info	23.86	5.7	85	0
Male	42.0	0	0	never	33.64	4.8	145	0
Female	32.0	0	0	never	27.32	5.0	100	0
Female	53.0	0	0	never	27.32	6.1	85	0
Female	54.0	0	0	former	54.7	6.0	100	0
Female	78.0	0	0	former	36.05	5.0	130	0
Female	67.0	0	0	never	25.69	5.8	200	0
Female	76.0	0	0	No Info	27.32	5.0	160	0
Male	78.0	0	0	No Info	27.32	6.6	126	0
Male	15.0	0	0	never	30.36	6.1	200	0
Female	42.0	0	0	never	24.48	5.7	158	0
Female	42.0	0	0	No Info	27.32	5.7	80	0
Male	37.0	0	0	ever	25.72	3.5	159	0
Male	40.0	0	0	current	36.38	6.0	90	0
Male	5.0	0	0	No Info	18.8	6.2	85	0
Female	69.0	0	0	never	21.24	4.8	85	0
Female	72.0	0	1	former	27.94	6.5	130	0
Female	4.0	0	0	No Info	13.99	4.0	140	0
Male	30.0	0	0	never	33.76	6.1	126	0
Male	67.0	0	1	not current	27.32	6.5	200	1
Male	40.0	0	0	former	27.85	5.8	80	0
Male	45.0	1	0	never	26.47	4.0	158	0
Male	43.0	0	0	never	26.08	6.1	155	0
Female	53.0	0	0	No Info	31.75	4.0	200	0

Pre Processing

Data Cleaning

- Handle missing values using imputation techniques or removal strategies.

- Remove duplicates and irrelevant entries.

Normalization

- Scale numerical features to standard ranges

- Ensures fair comparison among features during model training.

malization

```
y = df["diabetes"].values
x_df = df.drop(["diabetes"], axis = 1)
x = (x_df - np.min(x_df))/(np.max(x_df) - np.min(x_df))
x
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	gender_Female	gender_Male	smoking_history_current	smoking_history_non-smoker	smoking_history_past_
0	0.266667	0.000000	0.003333	0.083967	0.022000	0.466667	0.003333	0.000000	0.000000	0.003333	0
1	0.180000	0.000000	0.000000	0.091067	0.022000	0.266667	0.003333	0.000000	0.000000	0.003333	0
2	0.093333	0.000000	0.000000	0.091067	0.019000	0.526667	0.000000	0.003333	0.000000	0.003333	0
3	0.120000	0.000000	0.000000	0.078167	0.016667	0.516667	0.003333	0.000000	0.003333	0.000000	0
4	0.253333	0.003333	0.003333	0.067133	0.016000	0.516667	0.000000	0.003333	0.003333	0.000000	0
...	...	...	...	...	...	...	...	...	...	...	
99994	0.120000	0.000000	0.000000	0.082000	0.016000	0.483333	0.003333	0.000000	0.000000	0.003333	0
99996	0.006667	0.000000	0.000000	0.057900	0.021667	0.333333	0.003333	0.000000	0.000000	0.003333	0
99997	0.220000	0.000000	0.000000	0.092767	0.019000	0.516667	0.000000	0.003333	0.000000	0.000000	0
99998	0.080000	0.000000	0.000000	0.118067	0.013333	0.333333	0.003333	0.000000	0.000000	0.003333	0
99999	0.190000	0.000000	0.000000	0.074767	0.022000	0.300000	0.003333	0.000000	0.003333	0.000000	0

96128 rows × 11 columns

malization

```
y = df["diabetes"].values
x_df = df.drop(["diabetes"], axis = 1)
x = (x_df - np.min(x_df))/(np.max(x_df) - np.min(x_df))
x
```

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	gender_Female	gender_Male	smoking_history_current	smoking_history_non-smoker	smoking_history_past_
0	0.266667	0.000000	0.003333	0.083967	0.022000	0.466667	0.003333	0.000000	0.000000	0.003333	0
1	0.180000	0.000000	0.000000	0.091067	0.022000	0.266667	0.003333	0.000000	0.000000	0.003333	0
2	0.093333	0.000000	0.000000	0.091067	0.019000	0.526667	0.000000	0.003333	0.000000	0.003333	0
3	0.120000	0.000000	0.000000	0.078167	0.016667	0.516667	0.003333	0.000000	0.003333	0.000000	0
4	0.253333	0.003333	0.003333	0.067133	0.016000	0.516667	0.000000	0.003333	0.003333	0.000000	0
...	...	...	...	...	...	...	...	...	...	...	
99994	0.120000	0.000000	0.000000	0.082000	0.016000	0.483333	0.003333	0.000000	0.000000	0.003333	0
99996	0.006667	0.000000	0.000000	0.057900	0.021667	0.333333	0.003333	0.000000	0.000000	0.003333	0
99997	0.220000	0.000000	0.000000	0.092767	0.019000	0.516667	0.000000	0.003333	0.000000	0.000000	0
99998	0.080000	0.000000	0.000000	0.118067	0.013333	0.333333	0.003333	0.000000	0.000000	0.003333	0
99999	0.190000	0.000000	0.000000	0.074767	0.022000	0.300000	0.003333	0.000000	0.003333	0.000000	0

96128 rows × 11 columns

Pre Processing

Categorical Encoding

- Convert categorical variables using One-Hot Encoding or Label Encoding.
- Makes features machine-readable and suitable for modeling.

Categorical Encoding

```python
def perform_one_hot_encoding(df, column_name):
    dummies = pd.get_dummies(df[column_name], prefix=column_name)

    df = pd.concat([df.drop(column_name, axis=1), dummies], axis=1)

    return df

df = perform_one_hot_encoding(df, 'gender')

df = perform_one_hot_encoding(df, 'smoking_history')
```

Feature Selection

- Use statistical and model-based techniques
- Identify the most relevant predictors of diabetes.

Train-Test Split

- Split the dataset into training and testing subsets
- Ensures reliable evaluation of model performance.

```python
[12] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
```

Methodologies

- Data Preprocessing and Feature: The Kaggle dataset will be cleaned and prepared by handling missing values, normalizing numerical features, and encoding categorical variables. Feature selection techniques will be applied to identify the most relevant health-related factors influencing diabetes prediction.

- Model Implementation and Training: Various machine learning models, including Support Vector Machine (SVM), Decision Trees, Linear Regression, Decision Trees and Random Forest tree, will be implemented and trained on the dataset. Hyperparameter tuning and cross-validation will be conducted to optimize performance.

- Model Evaluation and Comparison: The trained models will be assessed using performance metrics (accuracy, recall and precision). A comparative analysis will be conducted to determine the most effective model for diabetes classification, providing insights for future healthcare applications

Models

- Support Vector Machine (SVM) can be used find the optimal hyperplane to separate diabetic and non-diabetic individuals, maximizing classification margin.

- Decision Trees will organizes data through a hierarchical structure, allowing for intuitive and interpretable classification decisions.

- Linear Regression models the relationship between input features and diabetes occurrence by fitting a line through data points, useful when a linear correlation exists.
- KNeighborsClassifier will finds the most similar past patients and predicts the same outcome (diabetic or not).
- Random Forest Tree will learns complex relationships between features (e.g., insulin levels, glucose) and diabetes risk.

KNeighbors, SVC

- KNeighborsClassifier achieves high accuracy (0.95) and precision (0.87), its classifies most instances correctly and reliably predicts diabetes when it does, but the low recall (0.53) indicates that many diabetic cases are missed and may struggle with identifying all patients who have diabetes.’

- SVC has high accuracy (95%) and precision (1.0), but low recall (0.38), indicating it misses a significant number of diabetic patients (false negatives).

```
knn = KNeighborsClassifier().fit(x_train, y_train)
knn_predicted = knn.predict(x_test)

knn_accuracy = knn.score(x_test, y_test)
knn_recall = recall_score(y_test, knn_predicted)
knn_precision = precision_score(y_test, knn_predicted)
print(f"Accuracy:{knn_accuracy},recall:{knn_recall}, percision:{knn_precision}")
```

```
Accuracy:0.9523561843337147,recall:0.5343915343915344, percision:0.8799612778315585
```

```
svm = SVC().fit(x_train, y_train)
svm_predicted = svm.predict(x_test)

svm_accuracy = svm.score(x_test, y_test)
svm_recall = recall_score(y_test, svm_predicted)
svm_precision = precision_score(y_test, svm_predicted)
print(f"Accuracy:{svm_accuracy},recall:{svm_recall}, percision:{svm_precision}")
```

```
Accuracy:0.9455424945386456,recall:0.3844797178130511, percision:1.0
```

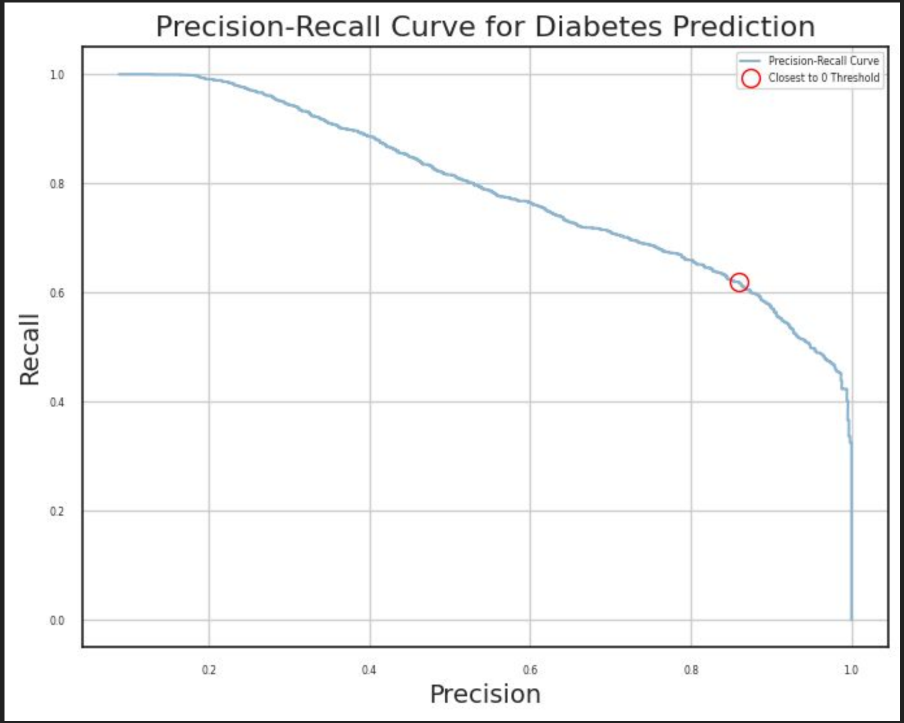
Logistic Regression

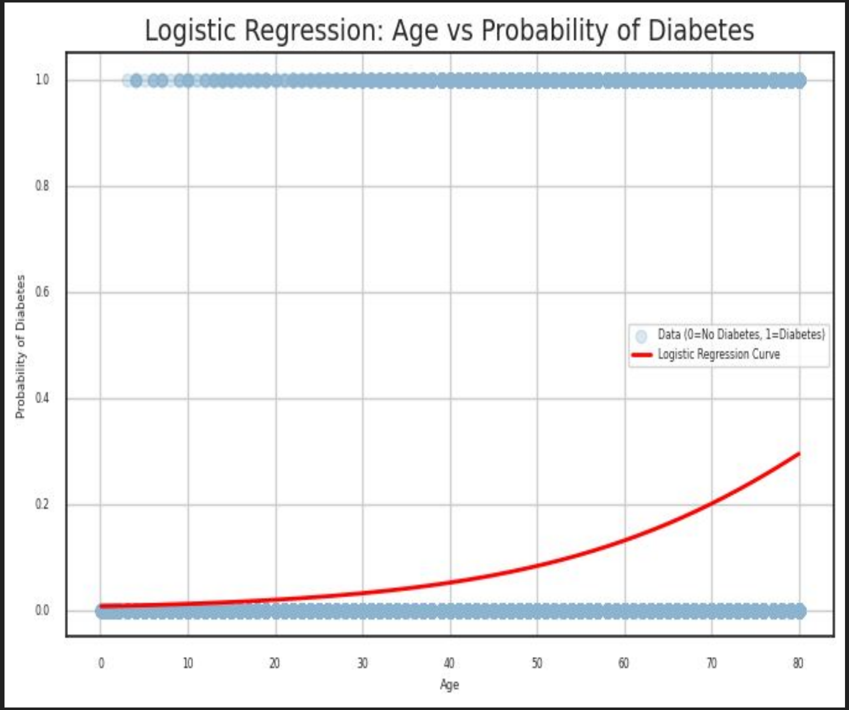
- Logistic Regression shows high accuracy (0.93) and precision (0.83), indicating it correctly classifies non-diabetic patients and has a good rate of correctly identifying positive cases, but struggles with a low recall (0.39), meaning it misses many diabetic patients.

```
lr_accuracy = lr.score(x_test, y_test)
lr_recall = recall_score(y_test, lr_predicted)
lr_precision = precision_score(y_test, lr_predicted)
print(f"Accuracy:{lr_accuracy},recall:{lr_recall}, percision:{lr_precision}")
```

```
Accuracy:0.9395610111307604,recall:0.3915343915343915, percision:0.8398486759142497
```

Logistic regression





Random Forest Classifier, Decision Tree

- RFC has high precision (0.94) and accuracy (0.96), Its effective at identifying non-diabetic patients but has a lower recall (0.66), missing about 34% of diabetic cases.
- The Decision Tree model achieved high accuracy (0.94), meaning it correctly predicted most diabetes cases overall.
- However, with a precision of 0.69 and recall of 0.72, it shows room for improvement in correctly identifying and confirming diabetic patients.

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier().fit(x_train, y_train)
dtc_predicted = dtc.predict(x_test)

dtc_accuracy =dtc.score(x_test, y_test)
dtc_recall = recall_score(y_test, dtc_predicted)
dtc_precision = precision_score(y_test, dtc_predicted)
print(f"Accuracy:{accuracy},recall:{recall}, percision:{precision}")
```

```
Accuracy:0.9673358993030271,recall:0.6696061140505585, percision:0.9452282157676348
```

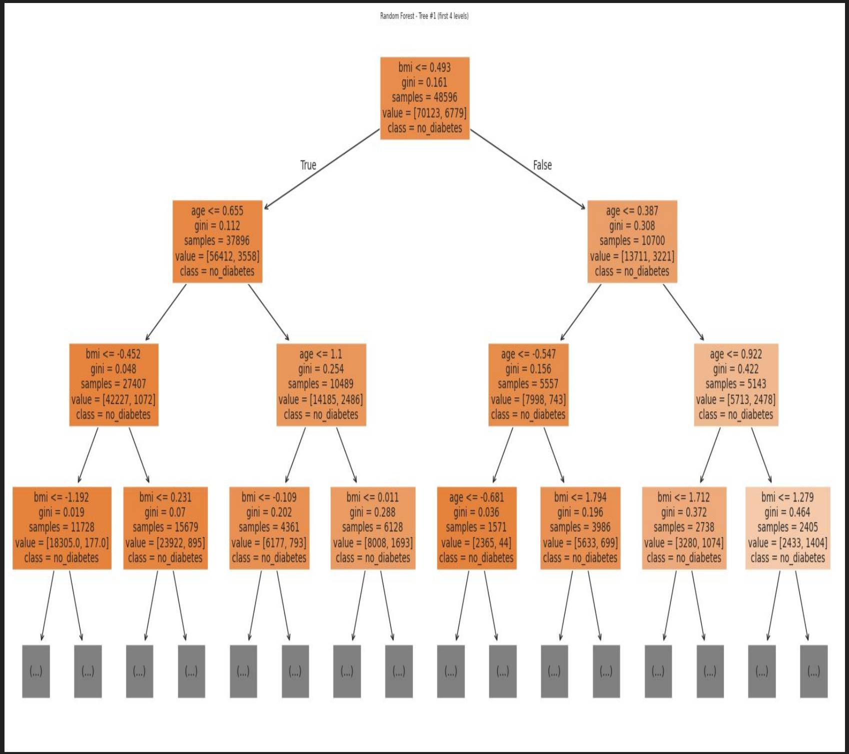
Random forest tree

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, recall_score, precision_score

    rfc = RandomForestClassifier().fit(x_train, y_train)
    rfc_predicted = rfc.predict(x_test)

    rfc_accuracy =rfc.score(x_test, y_test)
    rfc_recall = recall_score(y_test, rfc_predicted)
    rfc_precision = precision_score(y_test, rfc_predicted)
    print(f"Accuracy:{rfc_accuracy},recall:{rfc_recall}, percision:{rfc_precision}")
```

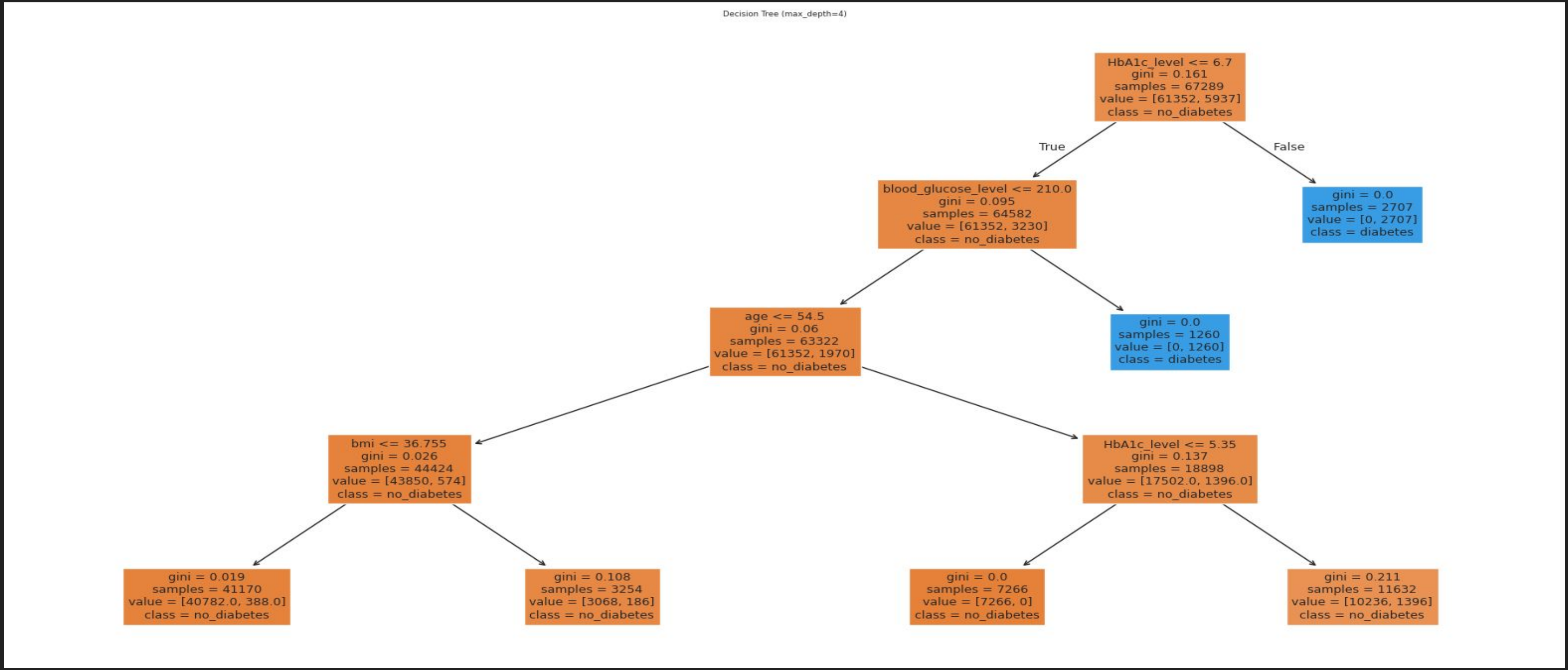
```
Accuracy:0.9673358993030271,recall:0.6696061140505585, percision:0.9452282157676348
```

Random Forest

	Feature	Importance
4	HbA1c_level	0.664588
5	blood_glucose_level	0.319929
0	age	0.013587
3	bmi	0.001896
1	hypertension	0.000000
2	heart_disease	0.000000
6	gender_Female	0.000000
7	gender_Male	0.000000
8	smoking_history_current	0.000000
9	smoking_history_non-smoker	0.000000



Decision Tree



Results

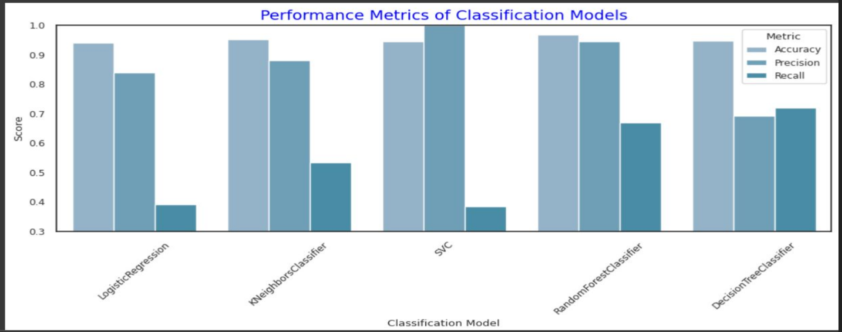
Logistic Regression has high precision but low recall meaning it is good at not over predicting but misses many cases

K-Nearest Neighbors has a balanced and have a better recall than Logistic Regression

Support Vector has a perfect precision at 1 meaning every diabetes prediction was correct but low recall meaning it missed many cases

Random Forest has the best metrics for performance as its balanced catching diabetes cases

Decision Tree has a lower precision than Random Forest but a high recall



1s completed at 9:07 AM

```
comparisonData.head(10)
```

	Accuracy	Precision	Recall
LogisticRegression	0.939561	0.839849	0.391534
KNeighborsClassifier	0.952356	0.879961	0.534392
SVC	0.945542	1.000000	0.384480
RandomForestClassifier	0.967336	0.945228	0.669606
DecisionTreeClassifier	0.947051	0.693265	0.720165

Future Findings

So when choosing a classification model for not just diabetes but other predictions it is advised to use Random Forest tree as its best and out performs other models as it is good for handling non-linear relationships leading to High accuracy, and handling of missing or dirty data.

The End