```
# Importing the Libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score
```

```
import pandas as pd
# Data Collection
data_set = pd.read_csv('/content/diabetes.csv')
data_set
```

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Next steps: Generate code with data_set | View recommended plots | New interactive sheet

```
# Data Preparation
y = data_set['Outcome']
x = data_set.drop('Outcome', axis =1)
```

```
# Standardization and Scaling
scaler = StandardScaler()
scaler.fit(x)
standardized_data = scaler.transform(x)
print (standardized_data)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

```
x = standardized_data
y = data_set['Outcome']
```

```
# Data Splitting
x_train, x_test, y_train, y_test, = train_test_split(x, y, test_size = 0.2, stratify= y, random_state=2)
```

```
# 1. Support Vector Machine (SVM)
# Import the necessary module
from sklearn import svm
# Initialize an empty dictionary called 'results' to store the accuracy scores.
results = {}
classifier = svm.SVC(kernel='linear')
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
results['SVM'] = accuracy_score(y_test, y_pred)

# 2. Random Forest Classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=2)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
results['Random Forest'] = accuracy_score(y_test, y_pred)

# 3. Logistic Regression
classifier = LogisticRegression(random_state=2)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
results['Logistic Regression'] = accuracy_score(y_test, y_pred)

# 4. Decision Tree Classifier
classifier = DecisionTreeClassifier(random_state=2)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
results['Decision Tree'] = accuracy_score(y_test, y_pred)

# 5. K-Nearest Neighbors (KNN)
classifier = KNeighborsClassifier(n_neighbors=5)  # Experiment with different 'n_neighbors'
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
results['KNN'] = accuracy_score(y_test, y_pred)

# 6. Neural Network (MLPClassifier)
classifier = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
results['Neural Network'] = accuracy_score(y_test, y_pred)

# Print Results
print("Accuracy Scores:")
for model, accuracy in results.items():
    print(f"{model}: {accuracy:.4f}")

# Find the best model
best_model = max(results, key=results.get)
print(f"\nBest Model: {best_model} with accuracy {results[best_model]:.4f}")
```

```
Accuracy Scores:
SVM: 0.7727
Random Forest: 0.7273
Logistic Regression: 0.7597
Decision Tree: 0.6948
KNN: 0.7208
Neural Network: 0.7403
```

```
Best Model: SVM with accuracy 0.7727
```

```
#Final Model Testing
input_data =(35,186,84,42,89,35,46,0.286)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
std_data = scaler.transform(input_data_reshaped)

prediction = classifier.predict(std_data)
print(prediction)

if(prediction[0]==0):
  print('The person is not diabetic\n')
else:
  print('The person is diabetic\n')
```

```
[0]
The person is not diabetic

/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature n
  warnings.warn(
```