In [128…
```
## Date - 24/10/2023
## Team ID -
## Project Title - AI Based Diabetes Perdiction Model
```

Importing Dependencies

In [129…
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import seaborn as sns
```

In [130…
```
# load data
data = pd.read_csv('diabetes.csv')
X = data.drop('Outcome', axis=1)
y = data['Outcome']
```

In [131…
```
# split 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)
```

Train the Model

Decision Tree Classifier

In [132…
```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[132]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [133…
```
# predict
```

In [134…
```
y_pred = model.predict(X_test)
```

In [135…
```
# Evaluate the model's performance
```

In [136…
```
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
Accuracy:  0.7337662337662337
Confusion Matrix:  [[76 23]
 [18 37]]
Classification Report:                precision    recall  f1-score   support

           0       0.81      0.77      0.79        99
           1       0.62      0.67      0.64        55

    accuracy                           0.73       154
   macro avg       0.71      0.72      0.72       154
weighted avg       0.74      0.73      0.74       154
```

Random Forest

In [137…
```
from sklearn.ensemble import RandomForestClassifier
```

In [138…
```
# Create an instance of Random Forest
model = RandomForestClassifier()
```

In [139…
```
# Fit the model to the training data
model.fit(X_train, y_train)
```

Out[139]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [140…
```
# Make predictions on the test data
y_pred = model.predict(X_test)
```

In [141…
```
# Evaluate the model's performance
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
Accuracy:  0.7467532467532467
Confusion Matrix:  [[78 21]
 [18 37]]
Classification Report:                precision    recall  f1-score   support

           0       0.81      0.79      0.80        99
           1       0.64      0.67      0.65        55

    accuracy                           0.75       154
   macro avg       0.73      0.73      0.73       154
weighted avg       0.75      0.75      0.75       154
```

K-Nearest Neighbors

In [142…
```
from sklearn.neighbors import KNeighborsClassifier
```

In [143…
```
# Create an instance of K-Nearest Neighbors
model = KNeighborsClassifier()
```

In [144…
```
# Fit the model to the training data
model.fit(X_train, y_train)
```

Out[144]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [145…
```
# Make predictions on the test data
y_pred = model.predict(X_test)
```

In [146…
```
# Evaluate the model's performance
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
Accuracy:  0.6623376623376623
Confusion Matrix:  [[70 29]
 [23 32]]
Classification Report:                precision    recall  f1-score   support

           0       0.75      0.71      0.73        99
           1       0.52      0.58      0.55        55

    accuracy                           0.66       154
   macro avg       0.64      0.64      0.64       154
weighted avg       0.67      0.66      0.67       154
```

Support Vector Machine

In [147…
```
from sklearn.svm import SVC
```

In [148…
```
# Create an instance of Support Vector Machine
model = SVC()
```

In [149…
```
# Fit the model to the training data
model.fit(X_train, y_train)
```

Out[149]:
```
▾ SVC
SVC()
```

In [150…
```
# Make predictions on the test data
y_pred = model.predict(X_test)
```

In [151…
```
# Evaluate the model's performance
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
```

```
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
Accuracy:  0.7662337662337663
Confusion Matrix:  [[87 12]
 [24 31]]
Classification Report:                precision    recall  f1-score   support

           0       0.78      0.88      0.83        99
           1       0.72      0.56      0.63        55

    accuracy                           0.77       154
   macro avg       0.75      0.72      0.73       154
weighted avg       0.76      0.77      0.76       154
```

Logistic Regression

In [152…
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [153…
```
# Split the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)
```

In [154…
```
# Create an instance of Logistic Regression
model = LogisticRegression()
```

In [155…
```
# Fit the model to the training data
model.fit(X_train, y_train)
```

```
c:\Users\Thiru-PC\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[155]:
```
▾ LogisticRegression
LogisticRegression()
```

In [156…
```
# Make predictions on the test data
y_pred = model.predict(X_test)
```

In [157…
```
# Evaluate the model's performance
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
print("Classification Report: ", classification_report(y_test, y_pred))
```

```
Accuracy:  0.7922077922077922
Confusion Matrix:  [[88  9]
 [23 34]]
Classification Report:                precision    recall  f1-score   support

           0       0.79      0.91      0.85        97
           1       0.79      0.60      0.68        57

    accuracy                           0.79       154
   macro avg       0.79      0.75      0.76       154
weighted avg       0.79      0.79      0.78       154
```