Practical 1

AIM :- Implement Linear Regression (Diabetes Dataset)

THEORY : -

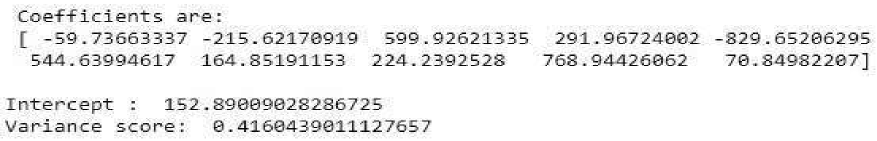
Logistic regression is a classification model in machine learning, extensively used in clinical analysis. It uses probabilistic estimations which helps in understanding the relationship between the dependent variable and one or more independent variables. Diabetes, being one of the most common diseases around the world, when detected early, may prevent the progression of the disease and avoid other complications. In this work, we design a prediction model, that predicts whether a patient has diabetes, based on certain diagnostic measurements included in the dataset, and explore various techniques to boost the performance and accuracy.

CODE / OUTPUT : -

```
# Import Dependencies
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets,linear_model,metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import seaborn as sns

# Load the diabetes dataset
diabetes=datasets.load_diabetes()

# X - feature vectors
# y - Target values
X=diabetes.data
y=diabetes.target

# splitting X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=1)


# Create linear regression object

lin_reg=linear_model.LinearRegression()


# Train the model using train and test data

lin_reg.fit(X_train,y_train)
```

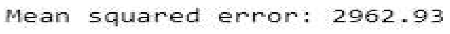
```
Out[6]: LinearRegression()
```

```
# Predict values for X_test data

predicted = lin_reg.predict(X_test)


# Regression coefficients

print('\n Coefficients are:\n',lin_reg.coef_)


# Intercept

print('\nIntercept : ',lin_reg.intercept_)


# variance score: 1 means perfect prediction

print('Variance score: ',lin_reg.score(X_test, y_test))
```



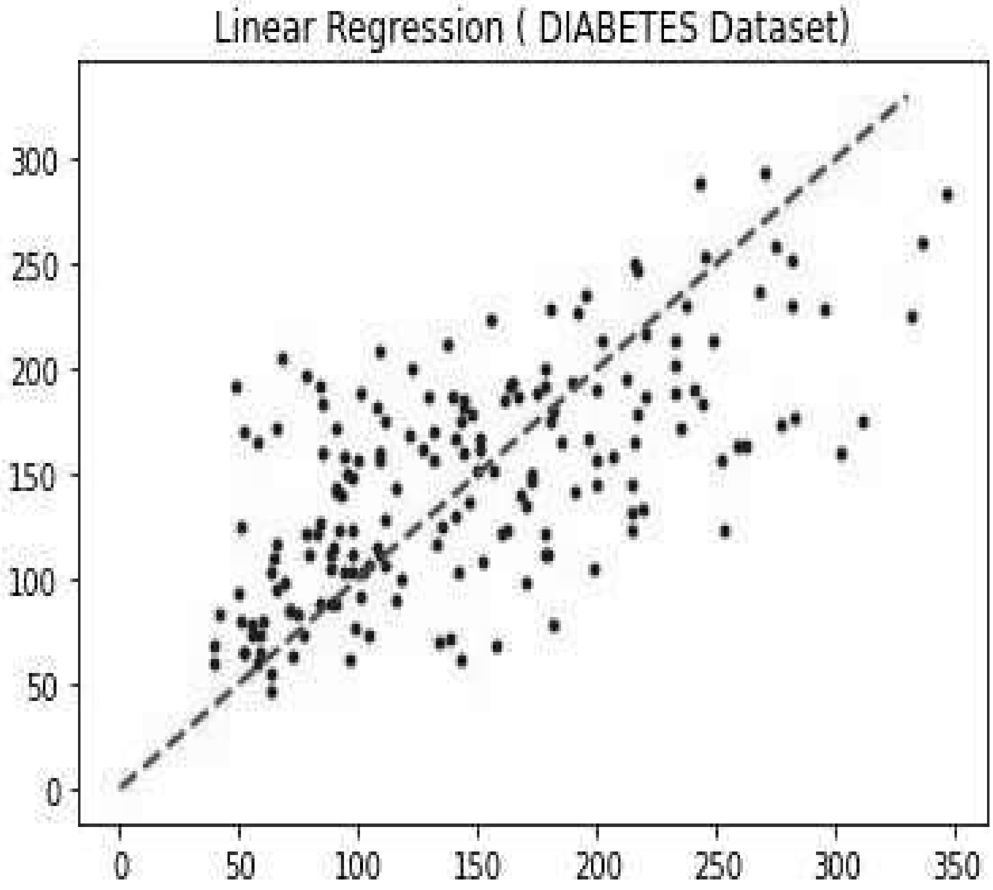
```
 Coefficients are:
 [ -59.73663337 -215.62170919  599.92621335  291.96724002 -829.65206295
  544.63994617  164.85191153  224.2392528   768.94426062   70.84982207]

Intercept :  152.89009028286725
Variance score:  0.4160439011127657
```

```
# Mean Squared Error

 print("Mean squared error: %.2f\n"

          % mean_squared_error(y_test, predicted))

# Original data of X_test

expected = y_test
```



```
Mean squared error: 2962.93
```

```
# Plot a graph for expected and predicted values

plt.title('Linear Regression ( DIABETES Dataset)')
plt.scatter(expected,predicted,c='b',marker='.',s=36)
plt.plot(np.linspace(0, 330, 100),np.linspace(0, 330, 100), '--r', linewidth=2)
plt.show()
```


Linear Regression (DIABETES Dataset)