Linear Regression for Diabetes Prediction

Step 1: Define our problem

We want to use the features in this kaggle healthcare dataset https://www.kaggle.com/datasets/nanditapore/healthcare-diabetes?resource=download to predict on the "Outcome" column 0/1 for diabetes prediction using Linear Regression.

Import relevant packages

In [71]:
```
#for data wrangling
import pandas as pd
#for scaling the data
from sklearn.preprocessing import StandardScaler
#for our dataset
import kagglehub
#for splitting the data
from sklearn.model_selection import train_test_split
#load our model
from sklearn.linear_model import LinearRegression
```

Step 2: Explore our data

In [29]:
```
#read our data into a dataframe
df = pd.read_csv('Healthcare-Diabetes.csv', encoding = 'latin-1')
#sneak peak of the data
df.head()
```

Out[29]:

	Id	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreel
0	1	6	148	72	35	0	33.6	
1	2	1	85	66	29	0	26.6	
2	3	8	183	64	0	0	23.3	
3	4	1	89	66	23	94	28.1	
4	5	0	137	40	35	168	43.1	

In [72]:
```
#describe the dataset to see rudimentary stats on its cols
df.describe()
```

Out[72]:

	Id	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
count	2768.000000	2768.000000	2768.000000	2768.000000	2768.000000	2768.000000
mean	1384.500000	3.742775	121.102601	69.134393	20.824422	80.127890
std	799.197097	3.323801	32.036508	19.231438	16.059596	112.301933
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	692.750000	1.000000	99.000000	62.000000	0.000000	0.000000
50%	1384.500000	3.000000	117.000000	72.000000	23.000000	37.000000
75%	2076.250000	6.000000	141.000000	80.000000	32.000000	130.000000
max	2768.000000	17.000000	199.000000	122.000000	110.000000	846.000000

In [77]:
```
#see most common occurances especially for our Outcome response var, this suggests
df["Outcome"].mode()
```

Out[77]:
```
0    0
Name: Outcome, dtype: int64
```

Step 3: Scale and Split our Data

In [46]:
```
#identify our X and y
X = df.drop(columns = ["Outcome"])
y = df["Outcome"]
scaler = StandardScaler()
#scale our data due to numerical column value differences
X_scaled = scaler.fit_transform(X)
```

In [38]:
```
#split our data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
                                                    random_state = 42, #for reprodu
                                                    test_size = 0.2, # 20% test siz
                                                    stratify = y) #keeps target sam
```

In [42]:
```
#validate shape
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
print("y Train shape:", y_train.shape)
print("y Test shape:", y_test.shape)
```

```
Train shape: (2214, 9)
Test shape: (554, 9)
y Train shape: (2214,)
y Test shape: (554,)
```

In [53]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[53]:

```
▾ LinearRegression
▸ Parameters
```

Step 4: Accuracy

In [65]:
```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import numpy as np
```

In [69]:
```
y_pred = lr.predict(X_test)
```

In [70]:
```
r2 = r2_score(y_test, y_pred)
print("R²:", r2)
mse = mean_squared_error(y_test, y_pred)
print("MSE:", mse)
mae = mean_absolute_error(y_test, y_pred)
print("MAE:", mae)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
```

```
R²: 0.288688905702674
MSE: 0.1606867419779891
MAE: 0.33743112306825396
RMSE: 0.40085750832183387
```

Conclusion

Why Linear Regression Struggles

Target is binary (0/1) → Linear regression isn't ideal for classification.

Outputs are continuous → Can produce predictions outside [0,1].

Assumptions of linearity are often violated with categorical/binary outcomes.