In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [6]:
```
from sklearn import model_selection
```

In [7]:
```
db=pd.read_csv("diabetes.csv")
```

In [24]:
```
db.describe()
```

Out[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [8]:
```
db
```

Out[8]:

	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

In [ ]:
```
#x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y)
```

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [10]:
```
x=db['Glucose']
y=db['Age']
```

In [11]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20)
```

In [13]:
```
print(x_train.shape)
```

```
(614,)
```

In [14]:
```
print(x_test.shape)
```

```
(154,)
```

In [15]:
```
print(y_train.shape)
print(y_test.shape)
```

```
(614,)
(154,)
```

In [16]:
```
from sklearn.linear_model import LinearRegression
```

In [17]:
```
alg1=LinearRegression()
```

In [20]:
```
alg1.fit(x_train,y_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-20-24557572384c> in <module>
----> 1 alg1.fit(x_train,y_test)

~\anaconda3\lib\site-packages\sklearn\linear_model\_base.py in fit(self, X, y, sample_weight)
    503
    504         n_jobs_ = self.n_jobs
--> 505         X, y = self._validate_data(X, y, accept_sparse=['csr', 'csc', 'coo'],
    506                                    y_numeric=True, multi_output=True)
    507

~\anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separately, **check_param
s)
    430                 y = check_array(y, **check_y_params)
    431             else:
--> 432                 X, y = check_X_y(X, y, **check_params)
    433             out = X, y
    434

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     71                           FutureWarning)
     72         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 73         return f(**kwargs)
     74     return inner_f
     75

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtyp
e, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numer
ic, estimator)
    794         raise ValueError("y cannot be None")
    795
--> 796     X = check_array(X, accept_sparse=accept_sparse,
    797                     accept_large_sparse=accept_large_sparse,
    798                     dtype=dtype, order=order, copy=copy,

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     71                           FutureWarning)
     72         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 73         return f(**kwargs)
     74     return inner_f
     75

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_large_sparse, d
type, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    618             # If input is 1D raise error
    619             if array.ndim == 1:
--> 620                 raise ValueError(
    621                     "Expected 2D array, got 1D array instead:\narray={}.\n"
    622                     "Reshape your data either using array.reshape(-1, 1) if "

ValueError: Expected 2D array, got 1D array instead:
array=[ 84  95  91  92 119 145  82  87 183 108  91 173  78 128 173 106 112 100
 131 100 146 190 104 186 176 102  80  88 118 114 125 105   0  95 108 154
 127 128 126 117  97 142 184 158  94 104  89  93  85  97  90  94 162 128
 128 163  80 102  76 139 116  95 129 180 101 175 136 180 164 151 133  84
 128 112 122  98 162 116 197 112 137  62  74 126 136 180 108  96 123 135
  99 102  87 128 151 194 103 100 130  99  99 102 131 134 147 108 101 159
 126 111 165 110  80 187 137 168  88 104 141  90 195  84  77  87 106 144
  83 164 170 181  95 123 143 106  68 109 165  90 138 115 109 125 145 119
 119  88  92 182 126  99 103 139 174 155  97 125 123  99 187 129 114 148
  71  80  90  97 133 139 108 130 169 148  99 154  95 117 158 129 109 102
 110 173 128 129 166 146 141 100 132 105 197 145  83 109 138 133 115  74
  92 145  92 126 154  71 112 131 115 196  89 129 102  77  99 111 176 119
  94 101 171 100 174 128 100 183 100 146 111  84  95 125  99 111 109 109
 117 105 105  44 144 141  87 114 161 124 117  83 100 168  79 106 151  84
 116 166 109 113 102 127 106 124 112 122 107 133 111   0  81  81  80 184
 127 162 144 116 167 152 148 112 111 130 173 120 162 124 117  72 180 179
 140 146 107  90 126 189  91 142  90  81 113 141  57  81  91 198 122 110
 128  94  97 152 129 168 130  85  88 113  99 106 112 106 163 106 114 134
 111 156 123 159 108  75 155  96 144 111 139 140 143 170 119 102 139 108
 112 129  78 167 162  68 154  92 125 142 124  92 139 124 130 105 108 195
 115 160 137 103  81  98 107  99 151 125 104  82 120 120 152  92  79 102
  99 146 103  94 189 141  91 122  99 100 109  91 125 107  85 151  74 129
  84 173 152 112 194 196 179 119 132 158 117 168 194 130 150 173  83 121
 100 116 113  93 122  99 136 119  97 157 101 107 104 113 111 155 129 125
 123 115 127 124 101  89 136  92 104 105 165 137 139 108  83  96  75 120
 136 156  90 134 197 125 102 123 135 107  88  82  61  95 126 107 187 116
 121 171  71 153 117 147 132 124  88 118  65 123 179  93 119 122  93 120
  85 105  85 187 120 107  78 147 109 136  89 178 107 106  93 193 140  84
 179 138 114 100 137  99 105  96 105 112  79  89  99 120 163 108 117 115
 103 146 147  90  95 181 183 102 116 114 111 125 181  96 134 138 136 129
  95 112  84 114 189  73 121 135 125 143 142  99  90 118 161 127 128  96
 123 108 158 101 140 150 107 146 122 129 144  90 118  84 122 109 129 106
 140 188 129 103 122  97 180 125 147 122 131 109 106 124 148 126  94 122
 188 107  67  95 111 154  81 117 181  56 115 155 105 122 179 109  71 112
  87 124].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it c
ontains a single sample.
```

In [21]:
```
alg1.predict(x_test)
```

```
---------------------------------------------------------------------------
NotFittedError                            Traceback (most recent call last)
<ipython-input-21-7b5db44553363> in <module>
----> 1 alg1.predict(x_test)

~\anaconda3\lib\site-packages\sklearn\linear_model\_base.py in predict(self, X)
    234             Returns predicted values.
    235         """
--> 236         return self._decision_function(X)
    237
    238     _preprocess_data = staticmethod(_preprocess_data)

~\anaconda3\lib\site-packages\sklearn\linear_model\_base.py in _decision_function(self, X)
    214
    215     def _decision_function(self, X):
--> 216         check_is_fitted(self)
    217
    218         X = check_array(X, accept_sparse=['csr', 'csc', 'coo'])

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     71                           FutureWarning)
     72         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 73         return f(**kwargs)
     74     return inner_f
     75

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_is_fitted(estimator, attributes, msg, all_or_any)
   1018
   1019     if not attrs:
-> 1020         raise NotFittedError(msg % {'name': type(estimator).__name__})
   1021
   1022

NotFittedError: This LinearRegression instance is not fitted yet. Call 'fit' with appropriate arguments before using
 this estimator.
```

In [ ]:
```
```