## **Regression Model**

```
Supervised learning has two type of model:

1)Regression
2)classification

1. Regression (CH-5)
Regression is continuous:

-> LR(linear regression)

-> Polynomial Regression

2. Classification (CH-6)
Classification is discrete:

-> KNN

-> Decision Tree
```

# LR(linear regression)

```
simple linear regression(SLR):
    y=mx+c
    where, y=dependent variable
        x=independent variable
        m=co-efficient
        c=intercept

multiple linear regression(MLR):
```

```
Step of Supervised learning process-

step1 : Define the problem
step2 : Collect and Prepare data
step3 : Data prepocessing
step4 : Feature Engineering
step5 : Select a model
step6 : Split the data in test/train
step7 : Train the model
step8 : Validate hyper parameters
step9 : Validate the model on the basis of tesing data
step10: Iterate the model
```

```
# Simple linear regression with the help of excel
```

# Simple linear regression model

```
In [5]: import pandas as pd
import numpy as np

df=pd.read_csv("placement_03-06.csv")
df
```

#### Out[5]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25
3	7.42	3.67
4	6.94	3.57
195	6.93	2.46
196	5.89	2.57
197	7.21	3.24
198	7.63	3.96
199	6.22	2.33

200 rows × 2 columns

## In [6]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
#
    Column
             Non-Null Count Dtype
             -----
     -----
 0
    cgpa
             200 non-null
                            float64
 1
    package 200 non-null
                            float64
dtypes: float64(2)
memory usage: 3.2 KB
```

```
In [7]: from sklearn.model_selection import train_test_split
          y=df['package']
          У
Out[7]: 0
                 3.26
                 1.98
          1
          2
                 3.25
                 3.67
          3
          4
                 3.57
                 . . .
          195
                 2.46
                 2.57
          196
          197
                 3.24
          198
                 3.96
                 2.33
          199
          Name: package, Length: 200, dtype: float64
In [11]: x=df.drop('package',axis=1)
Out[11]:
               cgpa
               6.89
            1
               5.12
               7.82
            3
               7.42
               6.94
          195
               6.93
          196
               5.89
          197
               7.21
          198
               7.63
          199
               6.22
          200 rows × 1 columns
In [12]: type(y)
Out[12]: pandas.core.series.Series
In [13]: type(x)
Out[13]: pandas.core.frame.DataFrame
In [14]: | y.shape
Out[14]: (200,)
```

```
In [15]: | x.shape
Out[15]: (200, 1)
In [49]: x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=
In [50]:
         print(y train.shape)
         print(y_test.shape)
         print(x_train.shape)
         print(x_test.shape)
         (160,)
         (40,)
         (160, 1)
         (40, 1)
In [51]: from sklearn.linear_model import LinearRegression
         lr=LinearRegression()
         model=lr.fit(x_train,y_train)
In [52]: print("m = ",model.coef )
         m = [0.58154877]
In [53]: print("c =" ,model.intercept_)
         c = -1.0859839580358024
In [54]: # y=mx+c
         y_pred=model.predict(x_test)
         print(y_pred)
         [2.9383335 4.36894346 3.18258398 1.89736121 3.49662031 3.35123312
          2.76968435 2.94996447 3.07208971 3.94441286 3.57222165 2.94996447
          2.75805338 2.64755911 3.67108494 3.2174769 3.97930579 2.90925606
          2.19395108 3.31052471 4.29915761 2.8918096 1.87409926 2.30444534
          3.62456104 2.12998071 3.9269664 2.36841571 1.5716939 2.06601035
          2.31026083 3.6885314 3.5024358 3.03719679 2.57195777 2.39167766
          3.170953
                    3.82228762 3.15932203 2.94414898]
```

## Out[55]:

	actual	predicted
58	3.09	2.938333
40	4.02	4.368943
34	3.42	3.182584
102	1.37	1.897361
184	3.14	3.496620
198	3.96	3.351233
95	2.79	2.769684
4	3.57	2.949964
29	3.49	3.072090
168	3.52	3.944413
171	3.76	3.572222
18	2.98	2.949964
11	2.60	2.758053
89	2.72	2.647559
110	3.76	3.671085
118	2.88	3.217477
159	4.08	3.979306
35	2.87	2.909256
136	2.10	2.193951
59	3.31	3.310525
51	3.79	4.299158
16	2.35	2.891810
44	1.86	1.874099
94	2.42	2.304445
31	3.89	3.624561
162	2.55	2.129981
38	4.36	3.926966
28	2.24	2.368416
193	1.94	1.571694
27	2.16	2.066010
47	3.26	2.310261
165	4.08	3.688531
194	3.67	3.502436
177	3.64	3.037197
176	3.23	2.571958

In [56]:

In [57]:

In [58]:

	actual	predicted
97	2.84	2.391678
174	2.99	3.170953
73	4.03	3.822288
69	2.94	3.159322
172	2.51	2.944149
# made mae prin	sklea mean_a t(mae)	solute er  y_act-y_ rn.metric bsolute_e
from		rn.metric
nnin	mean_s t(mse)	quared <u></u> er

0.7283345498058083

print(rs)

In [59]: from sklearn.metrics import r2\_score
 rs=r2\_score(y\_test,y\_pred)

# Multiple linear regression model-1

```
In [67]: import pandas as pd
    df=pd.read_csv("Advertising_03-06.csv")
    df
```

#### Out[67]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
				•••	
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

## In [68]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
```

#	Column	Non-Null Count	Dtype
0	Unnamed: 0	200 non-null	int64
1	TV	200 non-null	float64
2	Radio	200 non-null	float64
3	Newspaper	200 non-null	float64
4	Sales	200 non-null	float64

dtypes: float64(4), int64(1)
memory usage: 7.9 KB

```
In [69]: df.drop("Unnamed: 0",axis=1,inplace=True)
df
```

#### Out[69]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

```
In [70]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
### Columns Non Null Count Divisor
```

```
#
   Column
              Non-Null Count Dtype
              -----
0
   TV
              200 non-null
                             float64
              200 non-null
                             float64
1
   Radio
                             float64
2
   Newspaper 200 non-null
3
   Sales
              200 non-null
                             float64
```

dtypes: float64(4)
memory usage: 6.4 KB

```
In [71]: from sklearn.model_selection import train_test_split
    y=df['Sales']
y
```

```
Out[71]: 0
                  22.1
          1
                  10.4
          2
                  9.3
          3
                  18.5
          4
                  12.9
                  . . .
          195
                   7.6
          196
                   9.7
          197
                  12.8
          198
                  25.5
```

Name: Sales, Length: 200, dtype: float64

13.4

199

localhost:8888/notebooks/unit-5.ipynb#

```
In [72]: x=df.drop('Sales',axis=1)
x
```

#### Out[72]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4
195	38.2	3.7	13.8
196	94.2	4.9	8.1
197	177.0	9.3	6.4
198	283.6	42.0	66.2
199	232.1	8.6	8.7

200 rows × 3 columns

In [77]: from sklearn.model\_selection import train\_test\_split
x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.15,random\_state=

```
In [78]: print(x_train.shape)
    print(x_test.shape)
    print(y_train.shape)
    print(y_test.shape)

(170, 3)
```

(30, 3) (170,) (30,)

```
In [80]: from sklearn.linear_model import LinearRegression
lr=LinearRegression()
model=lr.fit(x_train,y_train)
print("m =" ,model.coef_)
print("c =" ,model.intercept_)

m = [ 0.04666462  0.18481678 -0.00121229]
c = 2.898009326357265

In [81]: y_pred=model.predict(x_test)
print(y_pred)

[21.8561195  16.43070013  7.61358579  17.81414167  18.64171793  23.81414579
  16.29442126  13.26019548  9.10013819  17.24141586  14.3795469  9.89375986
  17.34765633  16.79371444  14.88188303  15.48747719  12.40242285  17.2108581
  11.28920355  18.17106497  9.35301379  12.68796292  8.76659008  10.48401019
  11.33546207  15.00377232  9.8013108  19.48893945  18.43960804  17.16086278]
```

In [83]: diff=pd.DataFrame({"actual":y\_test,"predicted":y\_pred})
diff

## Out[83]:

	actual	predicted
58	23.8	21.856120
40	16.6	16.430700
34	9.5	7.613586
102	14.8	17.814142
184	17.6	18.641718
198	25.5	23.814146
95	16.9	16.294421
4	12.9	13.260195
29	10.5	9.100138
168	17.1	17.241416
171	14.5	14.379547
18	11.3	9.893760
11	17.4	17.347656
89	16.7	16.793714
110	13.4	14.881883
118	15.9	15.487477
159	12.9	12.402423
35	12.8	17.210858
136	9.5	11.289204
59	18.4	18.171065
51	10.7	9.353014
16	12.5	12.687963
44	8.5	8.766590
94	11.5	10.484010
31	11.9	11.335462
162	14.9	15.003772
38	10.1	9.801311
28	18.9	19.488939
193	19.6	18.439608
27	15.9	17.160863

```
In [84]: from sklearn.metrics import mean_absolute_error
    mae=mean_absolute_error(y_test,y_pred)
    print(mae)

    0.984560464503767

In [85]: from sklearn.metrics import mean_squared_error
    mse=mean_squared_error(y_test,y_pred)
    print(mse)

    1.8945245763596135

In [86]: from sklearn.metrics import r2_score
    rs=r2_score(y_test,y_pred)
    print(rs)
```

0.882855181551423

# Multiple linear regression model-2

```
In [91]: import pandas as pd
    df=pd.read_csv("insurance_03-06.csv")
    df
```

#### Out[91]:

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	ma <b>l</b> e	33.8	1	no	southeast	1725.55
2	28	ma <b>l</b> e	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	ma <b>l</b> e	28.9	0	no	northwest	3866.86
•••							
1333	50	ma <b>l</b> e	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83
1336	21	female	25.8	0	no	southwest	2007.95
1337	61	female	29.1	0	yes	northwest	29141.36

1338 rows × 7 columns

```
In [92]: df.info()
```

```
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
    Column
              Non-Null Count Dtype
_ _ _
     -----
                               ----
 0
              1338 non-null
                               int64
    age
 1
    sex
              1338 non-null
                               object
 2
    bmi
              1338 non-null
                               float64
 3
    children 1338 non-null
                               int64
 4
    smoker
              1338 non-null
                               object
 5
    region
              1338 non-null
                               object
 6
    expenses 1338 non-null
                               float64
dtypes: float64(2), int64(2), object(3)
```

<class 'pandas.core.frame.DataFrame'>

In [93]: df.drop(["sex","children","region"],axis=1,inplace=True)
df

#### Out[93]:

	age	bmi	smoker	expenses
0	19	27.9	yes	16884.92
1	18	33.8	no	1725.55
2	28	33.0	no	4449.46
3	33	22.7	no	21984.47
4	32	28.9	no	3866.86
1333	50	31.0	no	10600.55
1334	18	31.9	no	2205.98
1335	18	36.9	no	1629.83
1336	21	25.8	no	2007.95
1337	61	29.1	yes	29141.36

memory usage: 73.3+ KB

1338 rows × 4 columns

```
In [97]: df1=pd.get_dummies(df,drop_first=True)
df1
```

#### Out[97]:

	age	bmi	expenses	smoker_yes
0	19	27.9	16884.92	1
1	18	33.8	1725.55	0
2	28	33.0	4449.46	0
3	33	22.7	21984.47	0
4	32	28.9	3866.86	0
1333	50	31.0	10600.55	0
1334	18	31.9	2205.98	0
1335	18	36.9	1629.83	0
1336	21	25.8	2007.95	0
1337	61	29.1	29141.36	1

1338 rows × 4 columns

29141.36

```
In [103]: from sklearn.model_selection import train_test_split
    y=df1['expenses']
y
Out[103]: 0    16884.92
```

```
1
         1725.55
2
         4449.46
3
        21984.47
4
         3866.86
1333
        10600.55
1334
         2205.98
1335
         1629.83
1336
         2007.95
```

1337

Name: expenses, Length: 1338, dtype: float64

```
In [104]: x=df1.drop('expenses',axis=1)
x
```

#### Out[104]:

	age	bmi	smoker_yes
0	19	27.9	1
1	18	33.8	0
2	28	33.0	0
3	33	22.7	0
4	32	28.9	0
1333	50	31.0	0
1334	18	31.9	0
1335	18	36.9	0
1336	21	25.8	0
1337	61	29.1	1

1338 rows × 3 columns

```
In [105]:
          print(type(x))
          print(type(y))
          <class 'pandas.core.frame.DataFrame'>
          <class 'pandas.core.series.Series'>
In [106]:
          print(x.shape)
          print(y.shape)
          (1338, 3)
          (1338,)
In [110]: x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.85,random_state
          print(x_train.shape)
          print(x_test.shape)
          print(y_train.shape)
          print(y_test.shape)
          (1137, 3)
          (201, 3)
          (1137,)
          (201,)
```

```
In [112]: y_pred=model.predict(x_test)
print(y_pred)
```

```
[ 4.66662368e+03
                  1.33010678e+04
                                   1.34653152e+04
                                                   1.28193039e+04
  1.04531758e+03
                  3.12026024e+04
                                   1.31016999e+04
                                                   1.21308741e+04
  3.76778485e+03
                  3.02539216e+04
                                   1.18854074e+04
                                                   1.72896696e+04
  8.86756871e+03
                  8.61286966e+03
                                   4.06122202e+03
                                                   1.05509036e+04
  4.29383875e+03
                  6.38320741e+03
                                   1.53904365e+04
                                                   1.51357374e+04
  1.25646049e+04
                  3.24170861e+04
                                   9.24596826e+03
                                                   9.94182153e+03
  2.71199666e+03
                  8.16435467e+03
                                   8.37108330e+03
                                                   1.15236009e+04
  7.49432680e+03
                  4.39165114e+03
                                   1.43900422e+04
                                                   5.90331509e+03
                  2.72951572e+04
  3.32255987e+04
                                   3.29597957e+04
                                                   1.00008958e+04
  3.12247474e+04
                  2.58702645e+04
                                   1.59459962e+04
                                                   3.36224001e+04
  6.22444920e+03
                  1.44712615e+04
                                   1.03183495e+04
                                                   1.55787005e+04
  4.05573278e+03
                  1.31718781e+04
                                   4.86411999e+03
                                                   2.95746614e+04
  7.53499914e+03
                  1.24538798e+04
                                   1.43734491e+04
                                                   1.22157738e+04
  2.26535321e+03
                  8.35817060e+03
                                   2.56229262e+04
                                                   1.07797145e+04
  3.37608222e+04
                  1.52114048e+04
                                   2.16198890e+03
                                                   6.75967273e+03
  7.26913354e+03
                  1.49696184e+04
                                   2.72693318e+04
                                                   3.38570492e+03
  1.58075741e+04
                  1.12098276e+04
                                   1.03404946e+04
                                                   1.09477678e+04
  1.24649429e+03
                  2.50193961e+04
                                   3.66992504e+04
                                                   3.23635637e+04
  2.59571964e+03
                  1.03552788e+04
                                   1.41316627e+04
                                                   3,45138782e+04
  3.16606358e+03
                  4.88445616e+03
                                   1.10954221e+04
                                                   9.81631220e+03
 -6.80435782e+02
                  1.34081125e+04
                                   1.01577825e+04
                                                   4.02248391e+03
                  3.31239805e+04
                                   7.45190830e+03
  3.30428239e+04
                                                   3.75667119e+04
  1.17838519e+04
                  9.63908936e+03
                                   3.02354569e+04
                                                   3.25611227e+04
  1.45838581e+04
                  1.11784503e+04
                                   6.18947609e+02
                                                   1.16325172e+04
  9.95105385e+03
                  1.49234568e+04
                                   1.51301855e+04
                                                   5.06535939e+03
  1.40855011e+04
                  2.64092311e+04
                                   2.78046180e+04
                                                   2.82050999e+04
  3.62138688e+04
                  2.72065771e+04
                                   1.27787162e+03
                                                   9.77015062e+03
  4.97116470e+03
                  1.22471511e+04
                                   6.12295643e+03
                                                   4.73124988e+03
  1.43475829e+03
                  1.79375524e+04
                                   3.23805056e+03
                                                   2.53296497e+03
  1.04751735e+04
                  1.28544243e+04
                                   1.01484875e+04
                                                   3.79729064e+03
  9.62804819e+03
                  1.22452796e+04
                                   8.10528039e+03
                                                   7.55159223e+03
  3.68930663e+04
                  1.18373743e+04
                                   1.08665485e+04
                                                   2.91058729e+04
  3.59480032e+04
                  1.16749357e+04
                                   2.90116782e+04
                                                   8.55329335e+01
  7.19533771e+03
                  3.21623870e+04
                                   9.21459093e+03 -8.59906805e+00
  1.56026762e+03
                                   7.85800479e+03
                                                   1.23117147e+04
                  5.28493804e+03
                                   2.91280179e+04
                                                   1.65348047e+04
  1.46650774e+04
                  8.45411145e+03
  1.36554508e+04
                  1.15568498e+04
                                   2.49235533e+03
                                                   9.48220269e+03
  4.31224069e+03
                  5.55448403e+03
                                   1.19887090e+04
                                                   5.04502322e+03
  1.38824529e+04
                  1.30629618e+04
                                   1.36886370e+04
                                                   7.95394564e+03
  1.19905805e+04
                  1.05472232e+04
                                   9.67233823e+03
                                                   5.36247695e+03
                  4.06066956e+04
  6.67851612e+03
                                   1.37182054e+04
                                                   4.76262722e+03
  7.32820782e+03
                  5.62640832e+03
                                   3.27548760e+04
                                                   1.17986362e+04
  1.16915915e+04
                  6.64345840e+03
                                   6.26137847e+03
                                                   6.78924121e+03
  3.31905409e+04
                  3.51322552e+04
                                   2.35580483e+03
                                                   7.03845104e+03
  5.61349562e+03
                  1.41039658e+04
                                   1.29820779e+03
                                                   1.12780715e+04
  1.34856514e+04
                  1.13612250e+04
                                   1.07170226e+04
                                                   1.24722817e+04
  2.14907619e+03
                  2.85299771e+04
                                   3.12545393e+03
                                                   1.50286300e+04
  6.79853622e+03
                  8.54088267e+03
                                   1.45930904e+04
                                                   3.89897958e+04
  3.05165810e+03
                  1.21511695e+03
                                   4.59101900e+03
                                                   8.31943249e+03
  7.05136374e+03
                  4.57249167e+03
                                   9.51177117e+03
                                                   9.37147760e+03
  9.98424003e+03]
```

```
In [113]: diff=pd.DataFrame({"actual":y_test,"predicated":y_pred})
diff
```

#### Out[113]:

	actual	predicated
559	1646.43	4666.623676
1087	11353.23	13301.067793
1020	8798.59	13465.315239
460	10381.48	12819.303931
802	2103.08	1045.317580
891	7243.81	7051.363739
414	2134.90	4572.491675
258	11520.10	9511.771173
538	8233.10	9371.477595
929	6289.75	9984.240030

201 rows × 2 columns

```
In [114]: from sklearn.metrics import mean_absolute_error
mae=mean_absolute_error(y_test,y_pred)
print(mae)
```

4019.341739524793

```
In [115]: from sklearn.metrics import mean_squared_error
    mse=mean_squared_error(y_test,y_pred)
    print(mse)
```

36946790.51390431

```
In [116]: from sklearn.metrics import r2_score
    rs=r2_score(y_test,y_pred)
    print(rs)
```

0.739443551121639

## Model-3

#### Out[1]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmi
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	N
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	N
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	N
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	N
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	N
296	city	2016	9.50	11.60	33988	Diesel	Dealer	٨
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	٨
298	city	2009	3.35	11.00	87934	Petrol	Dealer	N
299	city	2017	11.50	12.50	9000	Diesel	Dealer	N
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	N

301 rows × 9 columns

In [2]: car.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Car_Name	301 non-null	object
1	Year	301 non-null	int64
2	Selling_Price	301 non-null	float64
3	Present_Price	301 non-null	float64
4	Driven_kms	301 non-null	int64
5	Fuel_Type	301 non-null	object
6	Selling_type	301 non-null	object
7	Transmission	301 non-null	object
8	Owner	301 non-null	int64
dtyp	es: float64(2),	int64(3), objec	t(4)

memory usage: 21.3+ KB

In [3]: car.drop("Car\_Name",axis=1,inplace=True)
car

## Out[3]:

	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Own
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	

301 rows × 8 columns

## Out[4]:

	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Own
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	
				•••			•••	
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	

301 rows × 9 columns

## Out[5]:

	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner	Ag
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	1
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	1
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	1
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	1
296	9.50	11.60	33988	Diesel	Dealer	Manual	0	
297	4.00	5.90	60000	Petrol	Dealer	Manual	0	
298	3.35	11.00	87934	Petrol	Dealer	Manual	0	1
299	11.50	12.50	9000	Diesel	Dealer	Manual	0	
300	5.30	5.90	5464	Petrol	Dealer	Manual	0	

301 rows × 8 columns

## Out[8]:

	Selling_Price	Present_Price	Driven_kms	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol
0	3.35	5.59	27000	0	10	0	1
1	4.75	9.54	43000	0	11	1	0
2	7.25	9.85	6900	0	7	0	1
3	2.85	4.15	5200	0	13	0	1
4	4.60	6.87	42450	0	10	1	0
296	9.50	11.60	33988	0	8	1	0
297	4.00	5.90	60000	0	9	0	1
298	3.35	11.00	87934	0	15	0	1
299	11.50	12.50	9000	0	7	1	0
300	5.30	5.90	5464	0	8	0	1

301 rows × 9 columns

```
In [9]: from sklearn.model_selection import train_test_split
          y=car1['Selling_Price']
          У
 Out[9]: 0
                    3.35
          1
                    4.75
          2
                    7.25
          3
                    2.85
          4
                   4.60
                   . . .
          296
                    9.50
          297
                   4.00
          298
                   3.35
          299
                  11.50
          300
                    5.30
          Name: Selling_Price, Length: 301, dtype: float64
In [10]: x=car1.drop("Selling_Price",axis=1)
          Х
Out[10]:
                Present_Price Driven_kms Owner Age Fuel_Type_Diesel Fuel_Type_Petrol Selling_type_In
             0
                                   27000
                                              0
                                                                    0
                        5.59
                                                  10
                                                                                     1
                                   43000
             1
                        9.54
                                              0
                                                  11
                                                                                    0
                                                                    1
             2
                                                   7
                        9.85
                                    6900
                                              0
                                                                    0
             3
                        4.15
                                    5200
                                                  13
                                                                    0
                                              0
                                                                                     1
                                              0
                                                  10
             4
                        6.87
                                   42450
                                                                    1
                                                                                    0
                          ...
                                              ...
           296
                        11.60
                                   33988
                                              0
                                                   8
                                                                    1
                                                                                    0
           297
                        5.90
                                   60000
                                              0
                                                   9
                                                                    0
                                                                                     1
           298
                        11.00
                                   87934
                                              0
                                                  15
                                                                    0
                                                                                     1
                        12.50
                                                   7
                                                                                    0
           299
                                    9000
                                              0
                                                                    1
           300
                        5.90
                                    5464
                                              0
                                                   8
                                                                    0
                                                                                     1
          301 rows × 8 columns
In [11]:
          print(type(x))
          print(type(y))
          <class 'pandas.core.frame.DataFrame'>
          <class 'pandas.core.series.Series'>
In [12]:
          print(x.shape)
          print(y.shape)
           (301, 8)
           (301,)
```

```
In [13]: x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.85,random_state
         print(x_train.shape)
         print(x_test.shape)
         print(y train.shape)
         print(y_test.shape)
         (255, 8)
         (46, 8)
         (255,)
         (46,)
In [14]: from sklearn.linear_model import LinearRegression
         lr=LinearRegression()
         model=lr.fit(x_train,y_train)
         print("m =" ,model.coef_)
         print("c =" ,model.intercept_)
         m = \begin{bmatrix} 4.35302568e-01 & -5.74656638e-06 & 3.09388161e-01 & -3.92843790e-01 \end{bmatrix}
           2.25837305e+00 5.10872356e-01 -1.21081820e+00 -1.83023303e+00]
         c = 6.805277248730253
In [15]: y pred=model.predict(x test)
         y_pred
Out[15]: array([ 7.78693357, 2.99462971, -0.50679037, 4.18535814, 0.52057204,
                  5.76800596, 1.91861631, 2.61742261, 7.67505414, 0.9729874,
                  8.07083232, 3.5281268, 4.88005217, 4.60002113, -2.04896272,
                  3.07278595, 7.92318893, 6.74538423, 6.87869784, 7.95766833,
                 4.25721114, 4.07722942, 11.29456886, 8.02126452, 9.47602518,
                  3.45113708, 3.87626941, 1.05764871, -0.56181678, -0.54720019,
                  0.03389518, -1.19234707, 4.26949393, 20.55308353, 18.62280514,
                 4.27026138, 3.52636571, 1.64807387, -0.02518064, 5.76668222,
                  7.97846986,
                              9.80428476, 0.40823392, 6.05156627, 5.83961851,
                 4.32779026])
```

In [17]: diff=pd.DataFrame({"actual":y\_test,"predicated":y\_pred})
diff

## Out[17]:

	actual	predicated		
285	7.40	7.786934		
248	4.00	2.994630		
150	0.50	-0.506790		
217	3.15	4.185358		
107	1.25	0.520572		
206	5.75	5.768006		
132	0.75	1.918616		
73	2.65	2.617423		
288	8.40	7.675054		
157	0.48	0.972987		
267	8.35	8.070832		
88	3.45	3.528127		
300	5.30	4.880052		
58	4.10	4.600021		
192	0.20	-2.048963		
177	0.35	3.072786		
11	6.85	7.923189		
230	6.15	6.745384		
224	5.11	6.878698		
29	7.45	7.957668		
27	6.00	4.257211		
293	3.25	4.077229		
78	5.25	11.294569		
12	7.50	8.021265		
85	2.50	9.476025		
18	3.25	3.451137		
298	3.35	3.876269		
139	0.60	1.057649		
180	0.30	-0.561817		
176	0.35	-0.547200		
182	0.30	0.033895		
197	0.16	-1.192347		
202	4.40	4.269494		
59	19.99	20.553084		
51	23.00	18.622805		

```
actual predicated
            89
                 4.75
                       4.270261
           246
                 3.75
                       3.526366
           120
                 1.05
                       1.648074
           191
                 0.20
                      -0.025181
           221
                 4.50
                       5.766682
                       7.978470
           256
                10.25
           250
                12.90
                       9.804285
           193
                 0.20
                       0.408234
                 4.60
                       6.051566
            4
            70
                 3.95
                       5.839619
           294
                 3.75
                       4.327790
In [18]: from sklearn.metrics import mean_absolute_error
          mae=mean_absolute_error(y_test,y_pred)
          print(mae)
          1.1919111738890504
          from sklearn.metrics import mean_squared_error
          mse=mean squared error(y test,y pred)
          print(mse)
          3.468310758909892
          from sklearn.metrics import r2 score
          rs=r2 score(y test,y pred)
          print(rs)
          0.8404169753392161
In [21]: x.columns
Out[21]: Index(['Present_Price', 'Driven_kms', 'Owner', 'Age', 'Fuel_Type_Diesel',
                  'Fuel_Type_Petrol', 'Selling_type_Individual', 'Transmission_Manual'],
                dtype='object')
In [27]: |y_pred=model.predict([[425000,100000,0,8,0,1,0,0]])
          print(y_pred)
          [185007.1902717]
         y pred=model.predict([[80000,425000,0,8,0,1,0,0]])
In [29]:
```

print(y\_pred)

[34825.93657286]

In [19]:

In [20]:

# **MODEL-4**

```
In [31]: import pandas as pd
import numpy as np
olm=pd.read_csv("olympic100m.csv")
olm
```

#### Out[31]:

	year	time
0	1896	12.00
1	1900	11.00
2	1904	11.00
3	1906	11.20
4	1908	10.80
5	1912	10.80
6	1920	10.80
7	1924	10.60
8	1928	10.80
9	1932	10.30
10	1936	10.30
11	1948	10.30
12	1952	10.40
13	1956	10.50
14	1960	10.20
15	1964	10.00
16	1968	9.95
17	1972	10.14
18	1976	10.06
19	1980	10.25
20	1984	9.99
21	1988	9.92
22	1992	9.96
23	1996	9.84
24	2000	9.87
25	2004	9.85
26	2008	9.69
27	2012	9.63
28	2016	9.81

```
In [32]: olm.info()
         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 29 entries, 0 to 28
         Data columns (total 2 columns):
               Column Non-Null Count Dtype
          _ _ _
           0
                       29 non-null
                                        int64
               year
           1
               time
                       29 non-null
                                        float64
         dtypes: float64(1), int64(1)
         memory usage: 592.0 bytes
In [33]: | from sklearn.model_selection import train_test_split
         y=olm['time']
         У
Out[33]: 0
                12.00
                11.00
         1
         2
                11.00
         3
                11.20
         4
                10.80
         5
                10.80
         6
                10.80
         7
                10.60
         8
                10.80
         9
                10.30
                10.30
         10
                10.30
         11
         12
                10.40
                10.50
         13
         14
                10.20
         15
                10.00
                 9.95
         16
         17
                10.14
                10.06
         18
         19
                10.25
         20
                 9.99
         21
                 9.92
                 9.96
         22
         23
                 9.84
         24
                 9.87
         25
                 9.85
         26
                 9.69
         27
                 9.63
                 9.81
         28
         Name: time, dtype: float64
```

### Out[37]:

```
year
   1896
 1 1900
 2 1904
 3 1906
 4 1908
  1912
  1920
 7 1924
 8 1928
  1932
   1936
11 1948
12 1952
13 1956
14 1960
15 1964
16 1968
17 1972
18 1976
  1980
19
   1984
20
21 1988
22 1992
23 1996
24 2000
25 2004
26 2008
27 2012
28 2016
```

```
In [38]:
```

```
print(type(x))
print(type(y))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

```
In [39]:
          print(x.shape)
          print(y.shape)
          (29, 1)
          (29,)
          x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.85,random_state
In [40]:
          print(x_train.shape)
          print(x_test.shape)
          print(y_train.shape)
          print(y_test.shape)
          (24, 1)
          (5, 1)
          (24,)
          (5,)
In [41]: from sklearn.linear_model import LinearRegression
          lr=LinearRegression()
          model=lr.fit(x_train,y_train)
          print("m = " ,model.coef_)
print("c = " ,model.intercept_)
          m = [-0.0129917]
          c = 35.764216687315766
In [42]: y_pred=model.predict([[2024]])
          print(y_pred)
          [9.46902066]
In [ ]:
```