PROGRAM

 # Import Libraries

```
import pandas as pd
import numpy as np
import warnings
warnings. Simplefilter ("ignore")
```

 # Load the Dataset

```
df = pd. read_csv("C:\\Users\\Documents\\Housing.csv")
df
```

Output

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	NaN	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	7.88	11.9

```
df.info ()
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

df. describe ()

Output

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.000000	506.000000	506.000000	506.000000	506.000000	486.0
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.795043	9.549407	408.237154	18.455534	356.674032	12.71
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.105710	8.707259	168.537116	2.164946	91.294864	7.15
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.73
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.100175	4.000000	279.000000	17.400000	375.377500	7.12
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.207450	5.000000	330.000000	19.050000	391.440000	11.43
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.188425	24.000000	666.000000	20.200000	396.225000	16.95
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.97

 # Explore and preprocess the data

df. isnull (). sum ()

Output

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

 # Replace the null values

```
from sklearn. impute import SimpleImputer
numeric_columns = ['CRIM', 'ZN','INDUS', 'CHAS', 'AGE', 'LSTAT']
df[numeric_columns] =
SimpleImputer(strategy='mean').fit_transform(df[numeric_columns])
df[numeric_columns]
```

Output

	CRIM	ZN	INDUS	CHAS	AGE	LSTAT
0	0.00632	18.0	2.31	0.0	65.200000	4.980000
1	0.02731	0.0	7.07	0.0	78.900000	9.140000
2	0.02729	0.0	7.07	0.0	61.100000	4.030000
3	0.03237	0.0	2.18	0.0	45.800000	2.940000
4	0.06905	0.0	2.18	0.0	54.200000	12.715432
...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	69.100000	12.715432
502	0.04527	0.0	11.93	0.0	76.700000	9.080000
503	0.06076	0.0	11.93	0.0	91.000000	5.640000
504	0.10959	0.0	11.93	0.0	89.300000	6.480000
505	0.04741	0.0	11.93	0.0	68.518519	7.880000

506 rows × 6 columns

Split the data into features (x)

x = df. iloc [:, :-1]

x

Output

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.200000	4.0900	1	296	15.3	396.90	4.980000
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.900000	4.9671	2	242	17.8	396.90	9.140000
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.100000	4.9671	2	242	17.8	392.83	4.030000
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.800000	6.0622	3	222	18.7	394.63	2.940000
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.200000	6.0622	3	222	18.7	396.90	12.715432
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.100000	2.4786	1	273	21.0	391.99	12.715432
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.700000	2.2875	1	273	21.0	396.90	9.080000
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.000000	2.1675	1	273	21.0	396.90	5.640000
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.300000	2.3889	1	273	21.0	393.45	6.480000
505	0.04741	0.0	11.93	0.0	0.573	6.030	68.518519	2.5050	1	273	21.0	396.90	7.880000

506 rows × 13 columns

Split the data into target variable (y)

y = df. iloc[:, -1]

y

Output

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

 # Further split the dataset into training and testing sets

from sklearn. model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split (x, y, random_state=0)

 # Intialize the Regularized Linear Regression model

from sklearn. linear_model import Ridge

 # Train a Ridge Regression model

 # Regularization strength, adjust as needed

alpha = 1.0

model = Ridge(alpha=alpha)

 # Train the Regularized Linear Regression model

model.fit (x_train, y_train)

Output

Ridge ()

 # Make predictions

y_pred = model. predict(x_test)

y_pred

Output

```
array([26.34156898, 22.20793785, 28.81921868, 11.47348568, 21.04408706,
       20.07024517, 19.92297674, 21.98467322, 19.4284628 , 19.78080258,
        4.16412665, 15.12289492, 17.09792351,  5.05900886, 38.91946091,
       32.92319545, 21.24037396, 36.10602752, 31.81994714, 23.86720827,
       25.23529707, 22.88616194, 20.79907381, 30.51097382, 22.69975385,
        7.89770805, 17.85693177, 18.79114443, 35.85644296, 20.64459013,
       17.22799563, 17.71776841, 19.12617187, 22.88297321, 28.53952139,
       19.82099868, 11.05567273, 23.90734107, 16.91704535, 14.40178292,
       25.90653761, 21.42101506, 23.7896341 , 13.97022376, 24.14528817,
       24.67733954, 19.26650784, 24.12468236, 11.09047876, 24.65867603,
       22.77081575, 19.26975343, 24.25327367, 31.6500543 , 12.84562266,
       22.28345771, 21.23812988, 16.08306186, 11.65101825, 22.23665653,
       18.60898999, 22.10455726, 32.35772086, 31.57758771, 17.12779565,
       32.87589474, 19.54848641, 19.28578883, 19.20291078, 24.09782582,
       21.94365685, 23.69076767, 30.63340076, 29.0446177 , 24.44329987,
        5.35195943, 37.02566061, 24.09822246, 27.67967687, 19.90269748,
       28.43908868, 18.85735312, 17.29445352, 37.82901475, 39.31585098,
       24.74164818, 25.00022374, 16.02274095, 26.54988516, 16.73956103,
       16.46269817, 13.41520098, 24.63687876, 30.64557876, 22.71383635,
       20.562259  ,  0.09343784, 25.64117227, 15.50351612, 17.61209673,
       25.92059772, 22.30531336, 32.48071611, 22.28998171, 27.57238428,
       23.45701196,  6.11176901, 14.10019865, 22.60348397, 29.02334184,
       31.92622327, 12.03657088, 19.53895825, 21.18421153, 12.1825899 ,
       23.82100592,  5.79235568, 19.29551109,  9.01036856, 45.15731178,
       30.5553796 , 17.34563703, 17.515943  , 22.17927199, 23.41151526,
       18.70788687, 34.97867185])
```

 # Evaluate the model

```
from sklearn. metrics import mean_squared_error
mse = mean_squared_error (y_test, y_pred)
print (f'Mean Squared Error: {mse}')
```

Output

Mean Squared Error: 32.12855445696262

 # Predict the new data

```
new_data = [0.02731, 0.0, 7.07, 0.0, 0.469, 6.421, 78.900000, 4.9671, 2, 242,
17.8, 396.90, 9.140000]
predictions = model. predict([new_data])
print(predictions)
```

Output

[24.67733954]

PROGRAM

```
# Import Libraries
import pandas as pd
import numpy as np
import warnings
warnings. Simplefilter ("ignore")

# Load the Dataset
df = pd. read_csv("C:\\Users\\Documents\\user_data.csv")
df
```

Output

	user_id	submission_count	problem_solved	contribution	country	follower_count	last_online_time_seconds	max_rating	rating	rank	regis
0	user_3311	47	40	0	NaN	4	1504111645	348.337	330.849	intermediate	
1	user_3028	63	52	0	India	17	1498998165	405.677	339.450	intermediate	
2	user_2268	226	203	-8	Egypt	24	1505566052	307.339	284.404	beginner	
3	user_480	611	490	1	Ukraine	94	1505257499	525.803	471.330	advanced	
4	user_650	504	479	12	Russia	4	1496613433	548.739	486.525	advanced	
...	...	...	...	...	...	...	...	...	...	...	
3566	user_2685	161	120	0	Bangladesh	42	1505409069	306.193	246.560	beginner	
3567	user_1548	41	30	0	NaN	0	1504026868	331.135	218.463	beginner	
3568	user_1929	58	51	0	NaN	0	1505552744	330.275	262.901	beginner	
3569	user_2772	148	137	0	NaN	2	1496606504	409.977	345.757	intermediate	
3570	user_2179	163	115	6	South Korea	40	1502074467	392.775	288.704	beginner	

3571 rows × 11 columns

```
df.info ()
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3571 entries, 0 to 3570
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   user_id                   3571 non-null   object
 1   submission_count          3571 non-null   int64
 2   problem_solved            3571 non-null   int64
 3   contribution              3571 non-null   int64
 4   country                   2418 non-null   object
 5   follower_count            3571 non-null   int64
 6   last_online_time_seconds  3571 non-null   int64
 7   max_rating                3571 non-null   float64
 8   rating                    3571 non-null   float64
 9   rank                      3571 non-null   object
 10  registration_time_seconds 3571 non-null   int64
dtypes: float64(2), int64(6), object(3)
memory usage: 307.0+ KB
```

```
df. describe ()
```

Output

	submission_count	problem_solved	contribution	follower_count	last_online_time_seconds	max_rating	rating	registration_time_seconds
count	3571.000000	3571.000000	3571.000000	3571.000000	3.571000e+03	3571.000000	3571.000000	3.571000e+03
mean	299.481098	267.894427	4.102492	46.690563	1.502680e+09	390.374392	350.165578	1.434961e+09
std	366.102887	344.139688	16.552256	211.494638	5.114850e+06	92.428788	106.592503	4.750758e+07
min	1.000000	0.000000	-64.000000	0.000000	1.484237e+09	303.899000	0.000000	1.264761e+09
25%	66.500000	53.000000	0.000000	4.000000	1.502691e+09	317.661000	279.243000	1.416323e+09
50%	169.000000	146.000000	0.000000	13.000000	1.505054e+09	355.791000	329.702000	1.449085e+09
75%	390.000000	349.000000	0.000000	40.000000	1.505551e+09	444.954000	413.417500	1.470379e+09
max	4570.000000	4476.000000	171.000000	10575.000000	1.505595e+09	983.085000	911.124000	1.484236e+09

 # Explore and preprocess the data

df. isnull (). sum ()

Output

```
user_id                     0
submission_count            0
problem_solved              0
contribution                0
country                  1153
follower_count              0
last_online_time_seconds    0
max_rating                  0
rating                      0
rank                        0
registration_time_seconds   0
dtype: int64
```

 # Encoding the categorical values

```
from sklearn. preprocessing import LabelEncoder
label_encoder = LabelEncoder ()
categorical_columns = ['country', 'rank']
df[categorical_columns]                                                    =
df[categorical_columns].apply(label_encoder.fit_transform)
df[categorical_columns]
```

Output

	country	rank
0	79	3
1	31	3
2	21	1
3	73	0
4	57	0
...	...	...
3566	5	1
3567	79	1
3568	79	1
3569	79	3
3570	62	1

3571 rows × 2 columns

 # Replace the null values

```
from sklearn. impute import SimpleImputer
numeric_columns = ['country']
df[numeric_columns] = SimpleImputer(strategy='mean').
fit_transform(df[numeric_columns])
df[numeric_columns]
```

Output

	country
0	79.0
1	31.0
2	21.0
3	73.0
4	57.0
...	...
3566	5.0
3567	79.0
3568	79.0
3569	79.0
3570	62.0

3571 rows × 1 columns

 # Drop the null values

```
df = df. drop ('user_id', axis =1)
df. head ()
```

Output

	submission_count	problem_solved	contribution	country	follower_count	last_online_time_seconds	max_rating	rating	rank	registration_time_seconds
0	47	40	0	79.0	4	1504111645	348.337	330.849	3	1466686436
1	63	52	0	31.0	17	1498998165	405.677	339.450	3	1441893325
2	226	203	-8	21.0	24	1505566052	307.339	284.404	1	1454267603
3	611	490	1	73.0	94	1505257499	525.803	471.330	0	1350720417
4	504	479	12	57.0	4	1496613433	548.739	486.525	0	1395560498

Split the data into features (x)

x = df. iloc [:, :-1]

x

Output

	submission_count	problem_solved	contribution	country	follower_count	last_online_time_seconds	max_rating	rating	rank
0	47	40	0	79.0	4	1504111645	348.337	330.849	3
1	63	52	0	31.0	17	1498998165	405.677	339.450	3
2	226	203	-8	21.0	24	1505566052	307.339	284.404	1
3	611	490	1	73.0	94	1505257499	525.803	471.330	0
4	504	479	12	57.0	4	1496613433	548.739	486.525	0
...	...	...	...	...	...	...	...	...	...
3566	161	120	0	5.0	42	1505409069	306.193	246.560	1
3567	41	30	0	79.0	0	1504026868	331.135	218.463	1
3568	58	51	0	79.0	0	1505552744	330.275	262.901	1
3569	148	137	0	79.0	2	1496606504	409.977	345.757	3
3570	163	115	6	62.0	40	1502074467	392.775	288.704	1

3571 rows × 9 columns

Split the data into target variable (y)

y = df. iloc[:, -1]

y

Output

```
0       1466686436
1       1441893325
2       1454267603
3       1350720417
4       1395560498
           ...
3566    1455055521
3567    1465142933
3568    1480086231
3569    1480262887
3570    1455975499
Name: registration_time_seconds, Length: 3571, dtype: int64
```

 # Further split the dataset into training and testing sets

```
from sklearn. model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split (x, y, random_state=0)
```

 # Intialize the Random Forest Regression model

```
from sklearn. ensemble import RandomForestRegressor
model = RandomForestRegressor (n_estimators=100, random_state=42)
```

 # Train the Random Forest Regression model

```
model.fit (x_train, y_train)
```

Output

```
RandomForestRegressor(random_state=42)
```

 # Make predictions

```
y_pred = model. predict(x_test)
y_pred
```

Output

```
array([1.45455496e+09, 1.43392280e+09, 1.39042627e+09, 1.45241261e+09,
       1.45483615e+09, 1.42245957e+09, 1.46665441e+09, 1.40576063e+09,
       1.45182356e+09, 1.40803296e+09, 1.44347665e+09, 1.45527652e+09,
       1.45833179e+09, 1.41039747e+09, 1.40353994e+09, 1.43335683e+09,
       1.45930505e+09, 1.48396266e+09, 1.45175827e+09, 1.46663339e+09,
       1.45883224e+09, 1.39686707e+09, 1.37510170e+09, 1.45583003e+09,
       1.43567271e+09, 1.46697796e+09, 1.42782285e+09, 1.35167334e+09,
       1.45055875e+09, 1.45104392e+09, 1.47095946e+09, 1.43119661e+09,
       1.39180426e+09, 1.45824164e+09, 1.44051569e+09, 1.46181146e+09,
       1.43774071e+09, 1.41057693e+09, 1.41982014e+09, 1.41032514e+09,
       1.46102952e+09, 1.44531362e+09, 1.44408272e+09, 1.43312909e+09,
       1.44923526e+09, 1.45480931e+09, 1.46493558e+09, 1.38739107e+09,
       1.46522130e+09, 1.34720321e+09, 1.44306016e+09, 1.43258882e+09,
       1.45973789e+09, 1.44872871e+09, 1.41464911e+09, 1.43173064e+09,
       1.47030693e+09, 1.41554286e+09, 1.45349140e+09, 1.45890839e+09,
       1.37963785e+09, 1.39024897e+09, 1.46494454e+09, 1.39503257e+09,
       1.36973396e+09, 1.46910568e+09, 1.45365958e+09, 1.44598027e+09,
       1.45992361e+09, 1.39585475e+09, 1.36215790e+09, 1.44300388e+09,
       1.42957886e+09, 1.45861093e+09, 1.39488787e+09, 1.46234938e+09,
```

 # Evaluate the model

```
from sklearn. metrics import mean_squared_error
mse = mean_squared_error (y_test, y_pred)
print (f'Mean Squared Error: {mse}')
```

Output

```
Mean Squared Error: 1269410250004051.2
```

 # Predict the new data

```
new_data = [161, 120, 0, 5.0, 42, 1505409069, 306.193, 246.560,1]
predictions = model. predict([new_data])
print(predictions)
```

Output

```
[1.45716732e+09]
```

PROGRAM

 # Import Libraries

```
import pandas as pd
import numpy as np
import warnings
```

warnings. Simplefilter ("ignore")

 # Load the Dataset

df = pd. read_csv("C:\\Users\\Documents\\50_Startups.csv")

df. head ()

Output

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

df.info ()

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

df. describe ()

Output

	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000

 # Explore and preprocess the data

df. isnull (). sum ()

Output

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

 # Encoding the categorical values

from sklearn import preprocessing

 # Using Label Encoder to convert the categorical values

le = preprocessing. LabelEncoder ()

state_encoded = le.fit_transform(data['State'])

data['State'] = state_encoded

data. head ()

Output

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	2	192261.83
1	162597.70	151377.59	443898.53	0	191792.06
2	153441.51	101145.55	407934.54	1	191050.39
3	144372.41	118671.85	383199.62	2	182901.99
4	142107.34	91391.77	366168.42	1	166187.94

 # Split the data into features (x)

x = df. iloc [:, :-1]

x

Output

	R&D Spend	Administration	Marketing Spend	State
0	165349.20	136897.80	471784.10	2
1	162597.70	151377.59	443898.53	0
2	153441.51	101145.55	407934.54	1
3	144372.41	118671.85	383199.62	2
4	142107.34	91391.77	366168.42	1

 # Split the data into target variable (y)

y = df. iloc[:, -1]

y

Output

```
0    192261.83
1    191792.06
2    191050.39
3    182901.99
4    166187.94
Name: Profit, dtype: float64
```

 # Further split the dataset into training and testing sets

from sklearn. model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split (x, y, random_state=355)

 # Intialize the Multivariate Regression model

```
from sklearn. Linear_model import LinearRegression
model = LinearRegression
```

 # Train the Random Forest Regression model

```
model.fit (x_train, y_train)
```

Output

```
LinearRegression ()
```

 # Make predictions

```
y_pred = model. predict(x_test)
y_pred
```

Output

```
array([126720.66150723,  84909.08961912,  98890.31854876,  46479.31240248,
       129113.18318813,  50968.88397762, 109015.01626803, 100893.57078084,
        97713.73821431, 113085.59056068])
```

 # Evaluate the model

```
from sklearn. metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error (y_test, y_pred)
print (f"Mean Squared Error: {mse}")
```

Output

Mean Squared Error: 80929465.49097784

```
mae = mean_absolute_error (y_test, y_pred)
print (f"Mean Absolute Error: {mae}")
```

Output

Mean Absolute Error: 6979.17574672139