Installing Libraries

```
!pip install numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (1.21
```

```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.3
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
```

```
print('prithivi Raj')
```

```
prithivi Raj
```

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

Data Collection and Processing

```
# Loading the csv data to a pandas DataFrame
```

```
gold_data=pd.read_csv("/content/sample_data/california_housing_test.csv")
gold_data.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.05	37.37	27.0	3885.0	661.0	1537.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-b8d59c0a8d95> in <module>()
----> 1 gold_data.head()

NameError: name 'gold_data' is not defined
```

SEARCH STACK OVERFLOW

```
gold_data.tail()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
2995	-119.86	34.42	23.0	1450.0	642.0	1258
2996	-118.14	34.06	27.0	5257.0	1082.0	3496
2997	-119.70	36.30	10.0	956.0	201.0	693
2998	-117.12	34.10	40.0	96.0	14.0	46
2999	-119.63	34.42	42.0	1765.0	263.0	753

```
gold_data.isnull.sum()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-7-2da4390a6a08> in <module>()
----> 1 gold_data.isnull.sum()

AttributeError: 'function' object has no attribute 'sum'
```

SEARCH STACK OVERFLOW

```
gold_data.isnull().sum()
#To find the missing value
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
```

```
median_house_value    0
dtype: int64
```

Double-click (or enter) to edit

```
gold_data.describe()
#To show the Statistical data
```

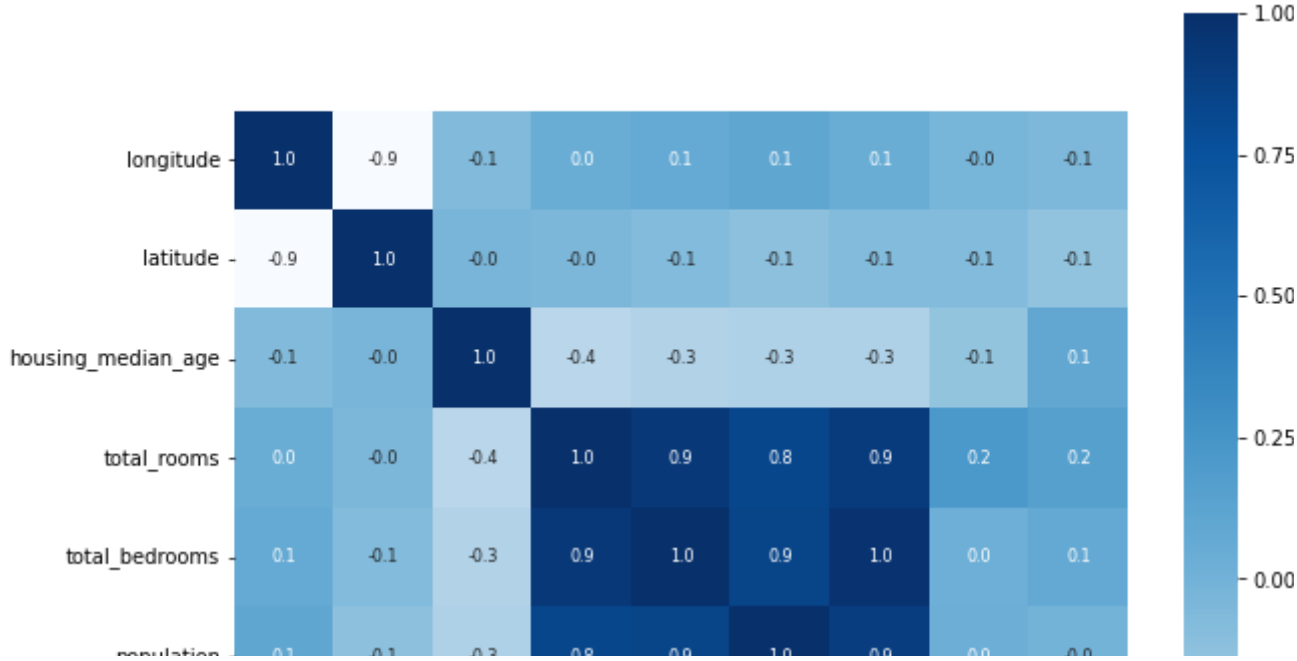
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	po
count	3000.000000	3000.00000	3000.000000	3000.000000	3000.000000	30
mean	-119.589200	35.63539	28.845333	2599.578667	529.950667	14
std	1.994936	2.12967	12.555396	2155.593332	415.654368	10
min	-124.180000	32.56000	1.000000	6.000000	2.000000	
25%	-121.810000	33.93000	18.000000	1401.000000	291.000000	7
50%	-118.485000	34.27000	29.000000	2106.000000	437.000000	11
75%	-118.020000	37.69000	37.000000	3129.000000	636.000000	17
max	-114.490000	41.92000	52.000000	30450.000000	5419.000000	119

Double-click (or enter) to edit

```
correlation=gold_data.corr()
```

```
plt.figure(figsize = (10,10))
sns.heatmap(correlation, cbar=True, square=True , fmt='.1f',annot=True,annot_kws={'size':8
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f993cf3dd90>
```

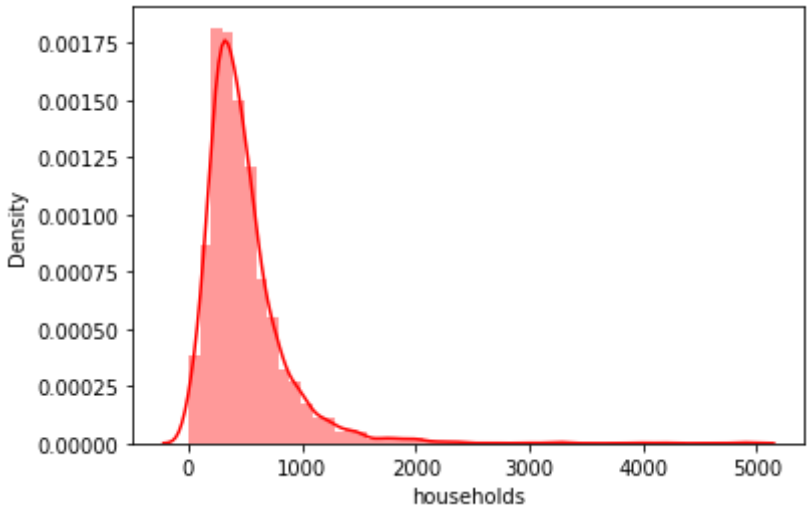



```
print(correlation['population'])
```

```
longitude             0.111572
latitude             -0.117318
housing_median_age   -0.299888
total_rooms           0.838867
total_bedrooms        0.856387
population            1.000000
households            0.895530
median_income         0.032361
median_house_value   -0.001192
Name: population, dtype: float64
```

```
sns.distplot(gold_data['households'],color='red')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f993804ff50>
```




```
X=gold_data.drop(['population','households'],axis=1)
```

```
Y=gold_data['households']

print(X)
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      median_income  median_house_value
0            6.6085            344700.0
1            3.5990            176500.0
2            5.7934            270500.0
3            6.1359            330000.0
4            2.9375             81700.0
...             ...                 ...
2995         1.1790            225000.0
2996         3.3906            237200.0
2997         2.2895             62000.0
2998         3.2708            162500.0
2999         8.5608            500001.0

[3000 rows x 7 columns]
```

```
print(Y)
```

```
0        606.0
1        277.0
2        495.0
3         11.0
4        237.0
         ...
2995     607.0
2996    1036.0
2997     220.0
2998      14.0
2999     260.0
Name: households, Length: 3000, dtype: float64
```

```
#Splitting of Training data and Test data
```

```
  File "<ipython-input-22-47f5a544a028>", line 2
    X_train, X_test , Y_train , Y_test = train_test_split(X, Y, test_size = 0.2, ranc

SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=2)

# MOdel Training
#Random Forest Regressor
regressor= RandomForestRegressor(n_estimators=100)



#Training the model
regressor.fit(X_train,Y_train)
```

```
RandomForestRegressor()
```

```
#MODEL EVALUATION
test_prediction=regressor.predict(X_test)
print(test_prediction)
```

```
[ 440.71  285.92  215.99  671.64  324.32  483.72  307.37   96.9   293.62
  953.49  536.68  343.79  635.66  557.67  143.19  451.77  220.01  420.01
 1071.49  263.27  383.13  555.87  344.27 1436.86  492.46  129.94  357.33
  147.59 1004.88  174.99  284.49  926.18  393.18  634.    475.5   467.65
  580.79  987.96  824.3   475.43 1196.6   648.6   828.56  780.02  405.24
  290.78  200.93  858.91  414.79  558.55  302.77  187.79  762.13  614.07
  741.74  358.04  334.06  706.55  236.94  102.27 3538.55  420.05  250.03
  393.45 1428.7   710.68  204.91  247.56 2004.02  752.79  810.06  225.32
  522.31  709.06  996.94 1323.44   79.19  703.17  300.18  274.04  313.81
  223.35  682.24  634.6   558.72  359.54  273.28  290.52  351.34 1143.55
  690.57 1201.29  413.62  558.14 1121.66  586.58   23.27  292.26  375.75
  343.12  695.18  753.26  199.36  612.78  426.59  143.67  373.97  121.
  490.92 1620.93  151.34  422.08  220.44  491.28   42.74  424.28  390.05
  369.07  475.31  460.68  347.15  852.4   426.75  489.69  388.94  242.81
 1386.45  381.25  293.31  450.58  231.47  472.13  459.62  193.02  407.87
   35.5   160.38  898.87  704.11  130.11 1825.71  352.37  235.72  572.77
  721.2   193.72  259.02  281.52   56.67  361.88  305.49  292.33  942.61
  340.03  903.35  248.59   11.25  874.47  353.59  294.56  442.75 1213.58
  832.83  620.09  635.75  144.14  634.6   296.47  361.84  506.71  234.82
  303.07  545.5   436.72  610.47  109.16  442.17  298.1   935.49  615.02
  266.53  317.23  425.2   459.42  385.    170.55  596.42  589.36  691.42
  402.03  347.55  198.44  429.96  511.4   276.16  604.61  406.08  174.07
  212.1   388.49    7.65  540.34  289.24  744.28  697.65  341.5   390.17
  501.7   527.57  239.06  472.75  383.47 1262.79  206.49  679.4   273.87
  522.21  497.82  411.25  975.53  324.59  634.66  963.34  501.94  203.41
  474.36  803.74  107.5   250.39 1016.75  210.36  533.12  492.1   288.97
   16.71  390.31  386.58  301.31  723.19  588.56  320.84  539.54  292.88
  483.56  210.4   360.03  703.41 1023.19   75.57  627.88 1284.64  460.03
  592.19  232.14  398.38  875.1   214.12  174.9   246.99  517.81  588.72
  252.37  968.9   292.34  351.92  575.51   14.98  196.2   627.36  501.08
  215.9   296.88 1070.16  733.78  187.48  294.64  276.3   856.21  173.16
  349.35  451.53 1075.46  688.59  429.15  217.68  255.03  384.91  742.48
  637.78  264.07  366.89  569.91  337.95  593.23  366.11  498.3   236.92
  576.2   111.14 1011.09  583.   1767.11  446.66   62.59 1020.01  124.97
 1870.48  306.45  579.64  270.03  636.49  272.81   48.38  295.26  219.02
  542.45  408.86  528.15  347.53  622.77  546.73  444.61  339.36  287.65
  324.17 1046.42  312.02  341.8   531.79  254.31  607.71  309.88    7.74
  720.    146.31  486.9   921.72  146.15  313.17   10.55  244.86  422.5
  887.75  619.76  568.28  300.16  289.46  514.31  461.25 1128.51  241.77
  388.25  795.99 1125.59  559.32  523.48  365.44  360.08  317.52  361.04
  364.72  469.54  495.02  211.01  463.07  404.38 1229.24  546.74  278.35
```

```
981.56  267.79 1176.32  331.18  188.86  582.96  364.89 1064.51  536.15
762.55  793.18  263.26  255.34  244.23  235.85  252.01  473.19  412.72
348.45  381.77  379.45  217.78 1052.56  304.06  593.09 1048.38  187.34
463.52  129.14  138.34  324.18  254.07  414.58  406.39  313.57  495.75
540.49 1411.91  342.04  282.95  280.11  398.61  359.92  569.89   80.18
705.47  334.47  589.68 1994.82  261.92  356.97  925.05  475.75  154.82
386.62  498.28  272.84  714.08  698.75  245.49  997.6   277.41  409.74
279.31  196.87  696.04  696.14  333.07  763.09  435.08  457.29  153.49
686.19  265.88  555.37  512.09  304.68  273.71  221.52  680.2   305.64
235.92  450.42  454.55  199.13  240.82  255.03  693.74  255.93   18.86
1111.21 567.86  305.85  866.67  390.21  205.39  349.05  326.83  390.63
234.23  615.86  282.14  373.43 1214.32  197.52 1399.74  712.73  350.58
235.56  414.87  427.78  532.47  193.63  468.78  624.97  671.9   718.56
725.75  450.53  485.07  403.75  267.91  684.41  347.17  447.97  224.18
102.63  535.59  991.    570.92 2702.58  277.57 1784.06  439.63  203.29
186.43  287.65  361.09  463.97  267.9   579.17  532.85  400.25  153.86
686.69  278.57  376.07  216.26  243.27  285.64  336.28  532.46  404.03
```

THESE ARE THE VALUES PREDICTED BY OUR MODEL

```
#Comparing the Predicted value and Actual Value ,we use metrics
# R Squared error
error_score = metrics.r2_score(Y_test, test_prediction)
print("R squared error : ", error_score)
```

```
R squared error :  0.9484579692960136
```

Double-click (or enter) to edit

Double-click (or enter) to edit

```
print("R squared error : ", error_score)
```

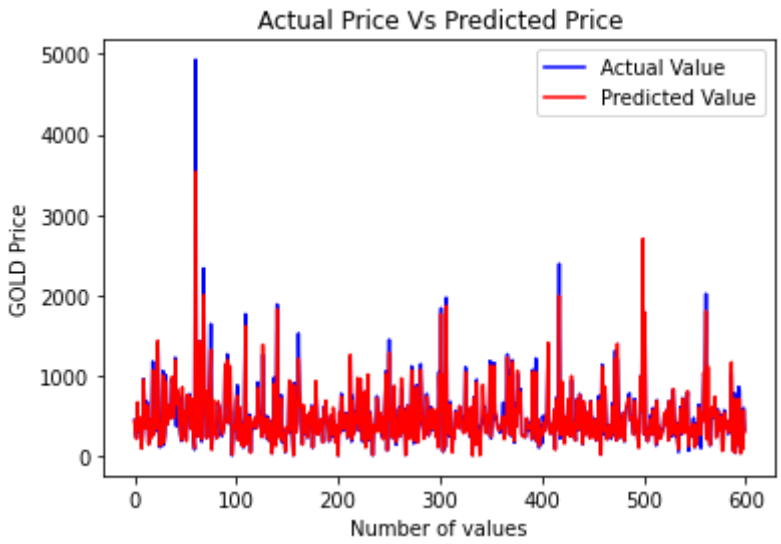
```
R squared error :  0.9484579692960136
```

Double-click (or enter) to edit

Comparing Actual Value and Predicted Value in a Plot

```
Y_test=list(Y_test)
```

```
plt.plot(Y_test,color='blue',label='Actual Value')
plt.plot(test_prediction,color='red', label='Predicted Value')
plt.title('Actual Price Vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('GOLD Price')
plt.legend()
plt.show()
```


Actual Price Vs Predicted Price
Actual Value
Predicted Value
GOLD Price
Number of values
0
1000
2000
3000
4000
5000
0
100
200
300
400
500
600