In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

In [2]:
```python
data=pd.read_csv("WQ.csv")
```

In [3]:
```python
data.head()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [4]:
```python
data.tail()
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

In [5]:
```python
data.describe()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

In [6]:
```python
data.info()
```

```
&lt;class 'pandas.core.frame.DataFrame'&gt;
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [7]:
```python
data.columns
```

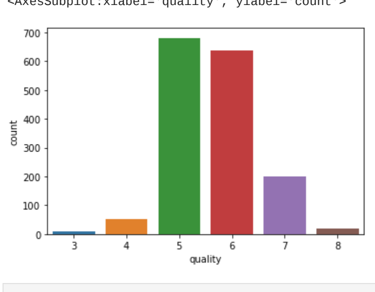
Out[7]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [8]:
```python
data['quality'].unique()
```

Out[8]:
```
array([5, 6, 7, 4, 8, 3], dtype=int64)
```

In [9]:
```python
sns.countplot(x='quality',data=data)
```

Out[9]:
```
&lt;AxesSubplot:xlabel='quality', ylabel='count'&gt;
```



In [11]:
```python
data['alcohol'].describe()
```

Out[11]:
```
count    1599.000000
mean       10.422983
std         1.065668
min         8.400000
25%         9.500000
50%        10.200000
75%        11.100000
max        14.900000
Name: alcohol, dtype: float64
```

In [12]:
```python
data['residual sugar'].describe()
```

Out[12]:
```
count    1599.000000
mean        2.538806
std         1.409928
min         0.900000
25%         1.900000
50%         2.200000
75%         2.600000
max        15.500000
Name: residual sugar, dtype: float64
```

In [13]:
```python
q1= data.quantile(0.25)
q2= data.quantile(0.75)
IQR=q2-q1
print(IQR)
```

```
fixed acidity            2.100000
volatile acidity         0.250000
citric acid              0.330000
residual sugar           0.700000
chlorides                0.020000
free sulfur dioxide     14.000000
total sulfur dioxide    40.000000
density                  0.002235
pH                       0.190000
sulphates                0.180000
alcohol                  1.600000
quality                  1.000000
dtype: float64
```

In [14]:
```python
print(data <(q1-1.5*IQR))| (data >(q2+1.5*IQR))
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0             False             False        False           False      False
1             False             False        False           False      False
2             False             False        False           False      False
3             False             False        False           False      False
4             False             False        False           False      False
...             ...               ...          ...             ...        ...
1594          False             False        False           False      False
1595          False             False        False           False      False
1596          False             False        False           False      False
1597          False             False        False           False      False
1598          False             False        False           False      False

      free sulfur dioxide  total sulfur dioxide  density     pH  sulphates  \
0                   False                 False    False  False      False
1                   False                 False    False  False      False
2                   False                 False    False  False      False
3                   False                 False    False  False      False
4                   False                 False    False  False      False
...                   ...                   ...      ...    ...        ...
1594                False                 False    False  False      False
1595                False                 False    False  False      False
1596                False                 False    False  False      False
1597                False                 False    False  False      False
1598                False                 False    False  False      False

      alcohol  quality
0       False    False
1       False    False
2       False    False
3       False    False
4       False    False
...       ...      ...
1594    False    False
1595    False    False
1596    False    False
1597    False    False
1598    False    False

[1599 rows x 12 columns]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in na_logical_op(x, y, op)
    301         #  (xint or xbool) and (yint or bool)
--> 302         result = op(x, y)
    303     except TypeError:

~\anaconda3\lib\site-packages\pandas\core\roperator.py in ror_(left, right)
     55 def ror_(left, right):
---> 56     return operator.or_(right, left)
     57

TypeError: unsupported operand type(s) for |: 'NoneType' and 'bool'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in na_logical_op(x, y, op)
    315             try:
--> 316                 result = libops.scalar_binop(x, y, op)
    317             except (

~\anaconda3\lib\site-packages\pandas\_libs\ops.pyx in pandas._libs.ops.scalar_binop()

ValueError: Buffer has wrong number of dimensions (expected 1, got 2)

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_2652/819519971.py in <module>
----> 1 print(data <(q1-1.5*IQR)) | (data >(q2+1.5*IQR))

~\anaconda3\lib\site-packages\pandas\core\ops\common.py in new_method(self, other)
     67         other = item_from_zerodim(other)
     68
---> 69         return method(self, other)
     70
     71     return new_method

~\anaconda3\lib\site-packages\pandas\core\arraylike.py in __ror__(self, other)
     72     @unpack_zerodim_and_defer("__ror__")
     73     def __ror__(self, other):
---> 74         return self._logical_method(other, roperator.ror_)
     75
     76     @unpack_zerodim_and_defer("__xor__")

~\anaconda3\lib\site-packages\pandas\core\frame.py in _arith_method(self, other, op)
   6864         self, other = ops.align_method_FRAME(self, other, axis, flex=True, level=None)
   6865
-> 6866         new_data = self._dispatch_frame_op(other, op, axis=axis)
   6867         return self._construct_result(new_data)
   6868

~\anaconda3\lib\site-packages\pandas\core\frame.py in _dispatch_frame_op(self, right, func, axis)
   6891             # i.e. scalar, faster than checking np.ndim(right) == 0
   6892             with np.errstate(all="ignore"):
-> 6893                 bm = self._mgr.apply(array_op, right=right)
   6894             return type(self)(bm)
   6895

~\anaconda3\lib\site-packages\pandas\core\internals\managers.py in apply(self, f, align_keys, ignore_failures, **kwargs)
    323             try:
    324                 if callable(f):
--> 325                     applied = b.apply(f, **kwargs)
    326                 else:
    327                     applied = getattr(b, f)(**kwargs)

~\anaconda3\lib\site-packages\pandas\core\internals\blocks.py in apply(self, func, **kwargs)
    379         """
    380         with np.errstate(all="ignore"):
--> 381             result = func(self.values, **kwargs)
    382
    383         return self._split_op_result(result)

~\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in logical_op(left, right, op)
    390         filler = fill_int if is_self_int_dtype and is_other_int_dtype else fill_bool
    391
--> 392         res_values = na_logical_op(lvalues, rvalues, op)
    393         # error: Cannot call function of unknown type
    394         res_values = filler(res_values)  # type: ignore[operator]

~\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in na_logical_op(x, y, op)
    323             ) as err:
    324                 typ = type(y).__name__
--> 325                 raise TypeError(
    326                     f"Cannot perform '{op.__name__}' with a dtyped [{x.dtype}] array "
    327                     f"and scalar of type [{typ}]"

TypeError: Cannot perform 'ror_' with a dtyped [bool] array and scalar of type [NoneType]
```

In [18]:
```python
data_out= data[~((data<(q1 - 1.5*IQR))| (data>(q2+1.5*IQR))).any(axis=1)]
data_out.shape
```

Out[18]:
```
(1179, 12)
```

In [19]:
```python
data_out
```

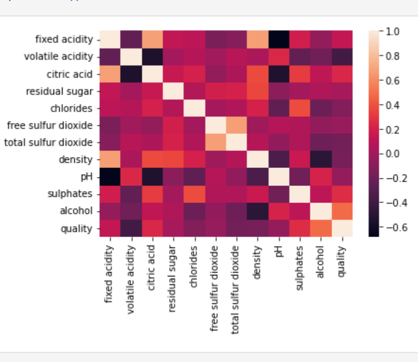
Out[19]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

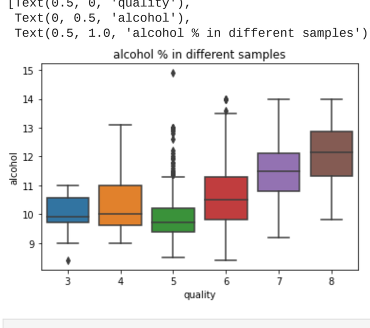
1179 rows × 12 columns

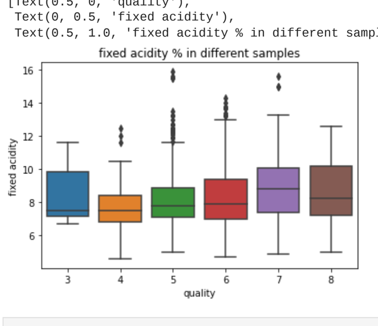
In [20]:
```python
correlation= data_out.corr()['quality'].drop('quality')
```

In [21]:
```python
sns.heatmap(data.corr())
plt.show()
```



[illegible]





[illegible]

```
       5.11499999, 5.19856301, 5.22167564, 5.15930339, 6.33123738,
       5.80081278, 5.0568108 , 5.94787774, 6.58386046, 6.32493852,
       4.84507443, 5.48619797, 5.80186036, 5.40637433, 5.68995625,
       5.59507948, 5.06544651, 5.41487779, 4.7249654 , 5.24536588,
       5.36232286, 5.49428148, 5.17775114, 5.45272283, 5.1170217 ,
       5.40357843, 6.09686498, 5.3462352 , 5.21862348, 5.95804421,
       5.66371381, 6.22070674, 5.63166059, 5.87312264, 6.30197092,
       5.58040881, 5.39105249, 5.80556937, 6.67727154, 6.4067045 ,
       5.7130713 , 5.30533251, 5.88099485, 5.71480393, 5.42444709,
       6.06942073, 5.52144259, 5.18229577, 5.77711526, 5.29208175,
       4.93161114, 6.48540857, 6.03860894, 5.67108966, 5.33858817,
       6.15166274, 6.29177186, 6.0334982 , 5.34189755, 6.23187009,
       4.7770051 , 5.50353636, 5.07288293, 6.1224935 , 5.59439803,
       5.49734441, 6.69186847, 5.07142699, 5.38242984, 5.47184162,
       5.22222483, 6.20864485, 5.91067653, 5.62479064, 5.7473586 ,
       5.3376445 , 5.06409375, 5.44259634, 5.26682364, 5.91219704,
       5.99975839, 6.15869422, 5.78544352, 4.87978546, 5.30068682,
       6.54719292, 5.27712555, 5.21735914, 6.49586406, 5.565757  ,
       6.17370377, 5.29701715, 5.59439803, 5.65847031, 5.97931   ,
       5.82112969, 5.19584919, 5.74960985, 5.76451145, 5.11777694,
       5.63189014, 6.20769442, 5.99823048, 6.31278529, 5.08507425,
       5.49782736, 6.06301554, 5.52208089, 5.48854536, 5.123057  ,
       6.3315908 , 5.37432323, 5.11563441, 6.29557617, 5.82112868,
       5.70484926, 5.88094406, 5.56486018, 5.19255154, 5.42257746,
       5.91222656, 5.88990437, 5.02373589, 6.13097115, 6.61205723,
       5.26813794, 6.12203499, 5.20645014, 5.3308828 , 5.81184852,
       5.52962273, 6.33402788, 5.54709807, 5.21774798, 5.09559314,
       6.32166859, 5.82813979, 5.81184852, 5.41631572, 6.48925591,
       6.30586651, 6.53188067, 5.36644261, 4.85142777, 6.10493763,
       4.91417356, 5.93883901, 6.00391823, 5.24391703, 5.2798985 ,
       5.39925211, 5.36644261, 5.4715931 , 4.93517947, 5.00186494,
       5.36615344, 5.67726568, 5.52898731, 5.28344018, 5.41943571,
       5.11515271, 5.69200372, 5.60001224, 5.22805622, 5.57676647,
       6.19911103, 5.50916455, 5.09564815, 6.19072202, 5.7837761 ,
       5.58709219, 6.05877138, 5.74736926, 5.42585456, 5.49320316,
       5.57296159, 6.25837569, 5.80913109, 5.79517626, 5.6482355 ,
       5.63166059, 5.61299893, 5.87294728, 5.86576571, 5.47204517,
       6.30236084, 5.911374  , 6.18795081, 6.57969079, 4.93523782,
       5.33273951, 5.67512976, 5.06908435, 5.5257644 , 5.9342683 ,
       6.00147884, 6.52767934, 5.30979772, 5.70927791, 5.63771522,
       5.50036196, 5.11515271, 5.95457555, 6.513226  , 5.61069356,
       5.81275986, 5.3609022 , 5.18382234, 5.40972513, 5.50623452,
       4.96860262, 5.38691885, 5.61091253, 5.12946122, 5.65092254,
       5.56953658, 5.2703363 , 5.6961587 , 5.2581744 , 5.65462698,
       5.47251363, 5.23852864, 5.23037638, 5.68512109, 5.4070197 ,
       5.68995625, 6.27692182, 6.163134  , 5.52404369, 5.47822543,
       5.54568711, 5.56245368, 6.50735218, 6.56646063, 5.17293625,
       5.77752669, 5.14824165, 5.20604932, 5.65052075, 5.50353636,
       5.06409375, 6.04914359, 4.84100642, 5.36644261, 5.101553  ,
       5.67108966, 5.14483778, 6.15869422, 5.16547531, 5.54754074,
       5.50714665, 5.49217966, 5.3736946 , 6.33383182, 5.06530238,
       5.24499396, 5.53686473, 5.55745727, 5.76892405, 5.80026999,
       6.25648323, 6.20753015, 4.9924918 , 6.26822657, 6.50303734,
       5.59996582, 6.09766492, 5.37989418, 5.67560393, 5.45674552,
       5.38435873, 5.87312264, 5.48063622, 5.74989395, 4.98759902,
       6.24437148, 6.58995156, 4.93830717, 5.21529197, 5.00244584,
       5.37766168, 6.15567245, 6.60369838, 5.31946648, 5.72184938,
       5.59317936, 5.25936283, 5.25730919, 5.47570108, 6.15567245,
       5.23204351, 5.30239928, 5.28335425, 6.03861458, 5.75085048,
       5.5289978 , 5.3663345 , 5.88489996, 5.0275512 , 5.3511309 ,
       6.30177007, 6.33336996, 5.48198848, 5.87294728, 5.56014018,
       5.12580074, 5.03018976, 5.08800544, 5.4079416 , 5.90271527,
       5.3154221 , 5.12654374, 6.29414372, 6.15991688, 5.0381891 ,
       6.29601829, 5.09248003, 6.1300031 , 5.18905152, 5.4349732 ,
       5.58631046, 5.58709219, 5.77305472, 5.13332863, 5.06329732,
       6.20411808, 5.68746015, 5.98696397, 5.61163635, 6.06629853,
       5.21774798, 5.20258581, 5.05495973, 5.87369086, 5.58778496,
       5.77084712, 6.307712  , 5.88199025, 6.31095042, 5.74441465,
       5.40357843, 5.09599607, 6.41443265, 5.78892634, 5.97512025,
       5.14962165, 6.16362365, 5.27820873, 6.42848296, 6.02515073,
       6.71785664, 5.85290174, 6.15653378, 6.10892543, 5.39393368,
       5.80485599, 5.93870605, 4.86335353, 5.95426304, 6.08819316,
       5.23204351, 6.30957888, 5.565757  , 5.21825697, 4.92310645,
       6.15991688, 5.8580258 , 5.62787409, 5.37401695, 6.4067045 ,
       6.28901085, 5.28349085, 5.23885629, 5.21970606, 5.07806046,
       5.29678052, 5.15833623, 5.23982877, 5.20390075, 4.78661624,
       6.14670094, 5.97636239, 5.88470611, 6.23299083, 5.03610653,
       5.76447027, 5.56857234, 5.51909511, 5.61618419, 5.82015356,
       5.74441465, 5.95804421, 5.37214771, 5.55409525, 5.8550351 ,
       5.62626975, 5.52760152, 5.70000066, 5.6482355 , 6.32068708,
       5.53128695, 5.59781054, 5.21756925, 5.40357843, 5.34189755,
       5.10472524, 6.01813045, 5.55745727, 6.08273954, 5.71274077,
       4.84100642, 5.64901139, 6.18892256, 6.15472394, 6.40297502,
       5.0323977 , 5.48492275, 5.26813794, 5.31868162, 5.3661849 ,
       5.46074367, 6.0100637 , 5.9796431 , 5.38691885, 6.07105301,
       4.91053182, 6.19072202, 5.92064592, 6.41493553, 5.98600093,
       5.96240259, 5.68973263, 5.68743537, 6.20706998, 5.75812996,
       6.11882856, 5.30332594, 6.62270262, 6.38798739, 5.46332282,
       5.50619161, 6.53188067, 5.18647495, 5.71242758, 5.44509131,
       5.40094065, 5.20645014, 5.15849111])
```

In [50]:
```python
test_pred=regression.predict(x_test)
test_pred
```

Out[50]:
```
array([5.61948116, 5.15676143, 5.35463723, 5.87369086, 5.41906975,
       6.19248853, 5.74035291, 5.64727915, 5.98671473, 6.00072399,
       5.27454777, 5.40945434, 5.57676647, 5.57838617, 5.45287962,
       5.92978555, 5.30334908, 5.9717136 , 5.22335787, 5.45272283,
       5.22298198, 5.80186036, 5.8708036 , 5.37242241, 5.20884814,
       5.69829008, 5.91933139, 5.50964271, 5.2742391 , 5.18914673,
       5.73773759, 6.15823912, 5.33843571, 5.94245806, 5.19951764,
       5.35949131, 6.02743914, 5.28680258, 5.77153256, 5.36644261,
       5.59386183, 5.87189867, 5.41081511, 5.2656031 , 5.21143201,
       5.56873032, 5.02663018, 5.86410748, 5.93883901, 5.65200469,
       5.37837239, 6.27574578, 5.49805162, 5.5188083 , 5.36706206,
       5.80936933, 5.69829008, 5.9635673 , 5.2148181 , 5.40225228,
       5.00599636, 6.23299083, 5.19099648, 4.97051039, 5.67512976,
       5.3308828 , 5.32298492, 6.73751649, 5.14000912, 4.66001182,
       5.09511008, 5.75344256, 5.55441525, 4.99722638, 5.2580626 ,
       5.47086312, 5.30546436, 5.20106422, 5.29261872, 5.38272235,
       5.30485161, 5.26000016, 5.90271527, 5.09848534, 6.00727672,
       5.59060136, 5.22941222, 5.57441106, 5.61163635, 6.4635574 ,
       5.02459147, 6.32527157, 6.13097115, 6.47640016, 5.30032349,
       5.22234124, 5.98636391, 5.12878767, 5.87131644, 5.20390075,
       6.38942355, 5.35984636, 5.26807669, 4.98279895, 5.47822543,
       5.14562594, 6.44812666, 6.69618458, 5.03610653, 5.52275875,
       5.73897817, 5.1568226 , 5.60552579, 5.17380791, 5.82797532,
       5.42257746, 6.6499719 , 5.03885866, 5.76574517, 6.08139394,
       5.58709219, 5.29443058, 5.82462713, 5.45958046, 5.18624952,
       5.6327017 , 5.99356431, 5.54707273, 5.76657853, 5.05672429,
       5.48819888, 6.16055292, 5.57102546, 5.40382215, 6.21584025,
       5.93120481, 5.23665437, 5.91067653, 5.1810682 , 5.84909792,
       5.80783531, 6.0263082 , 5.86891453, 5.0381891 , 5.46064463,
       5.82630789, 5.42634765, 5.24888097, 5.4240555 , 5.40780113,
       6.34651266, 5.38138031, 5.04780018, 5.83424454, 6.26648296,
       6.34417819, 5.51932518, 5.09741059, 6.27579257, 5.2480789 ,
       5.31421043, 6.42142059, 6.07204285, 5.86576571, 6.04284948,
       4.97966776, 5.92736963, 6.19072202, 5.20465451, 5.29130437,
       5.26000016, 5.58634102, 5.70161938, 5.73618272, 5.75600385,
       5.00515388, 5.34447523, 6.20260509, 5.02487969, 5.37147281,
       5.5900819 , 5.36488282, 5.8314728 , 5.09268074, 5.49175761,
       5.26000016, 6.00124977, 5.75081725, 5.49164003, 5.86372445,
       5.37978529, 5.88099485, 6.49990095, 5.21825697, 6.19041585,
       5.28447672, 6.05414271, 5.40127984, 6.09462497, 5.48302238,
       6.08139394, 5.3482185 , 5.2475974 , 6.59066616, 5.20636317,
       5.3823848 , 6.00292074, 5.81187172, 5.54870198, 5.74313205,
       5.09848534, 5.49582854, 5.11577369, 5.20884814, 5.65200469,
       5.07858342, 5.73221695, 5.98636391, 6.13439934, 6.34577646,
       5.17111584, 5.88029393, 6.29095732, 5.62668206, 5.11421755,
       5.06166028, 5.19768039, 5.85926314, 6.39795733, 5.12263143,
       5.14515606, 5.98671473, 5.36615344, 5.40225228, 5.05672429,
       6.02743914])
```

In [51]:
```python
train_rms= metrics.mean_squared_error(train_pred, y_train)**0.5
train_rms
```

Out[51]:
```
0.5724464562781634
```

In [53]:
```python
train_rms= metrics.mean_squared_error(test_pred, y_test)**0.5
train_rms
```

Out[53]:
```
0.5609706590975136
```

In [54]:
```python
predictData= np.round_(test_pred)
predictData
```

Out[54]:
```
array([6., 5., 5., 6., 5., 6., 6., 6., 6., 6., 5., 5., 6., 6., 5., 6., 5.,
       6., 5., 5., 5., 6., 6., 5., 5., 6., 6., 6., 5., 5., 6., 6., 5., 6.,
       5., 5., 6., 5., 6., 5., 6., 6., 5., 5., 5., 6., 5., 6., 6., 6., 5.,
       6., 5., 6., 5., 6., 6., 6., 5., 5., 5., 6., 5., 5., 6., 5., 5., 7.,
       5., 5., 5., 6., 6., 5., 5., 5., 5., 5., 5., 5., 5., 5., 6., 5., 6.,
       6., 5., 6., 6., 6., 5., 6., 6., 6., 5., 5., 6., 5., 6., 5., 6., 5.,
       5., 5., 5., 5., 6., 7., 5., 6., 6., 5., 6., 5., 6., 5., 7., 5., 6.,
       6., 6., 5., 6., 5., 5., 6., 6., 6., 6., 5., 5., 6., 6., 5., 6., 6.,
       5., 6., 5., 6., 6., 6., 6., 5., 5., 6., 5., 5., 5., 5., 6., 5., 5.,
       6., 6., 6., 6., 5., 6., 5., 5., 6., 6., 6., 6., 5., 6., 6., 5., 5.,
       5., 6., 6., 6., 6., 5., 5., 6., 5., 5., 6., 5., 6., 5., 5., 5., 6.,
       6., 5., 6., 5., 6., 6., 5., 6., 5., 6., 5., 6., 5., 6., 5., 5., 7.,
       5., 5., 6., 6., 6., 6., 5., 5., 5., 5., 6., 5., 6., 6., 6., 6., 5.,
       6., 6., 6., 5., 5., 5., 6., 6., 5., 5., 6., 5., 5., 5., 6.])
```

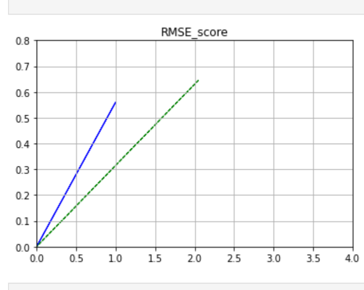
In [55]:
```python
print('mean absolute error:', metrics.mean_absolute_error(y_test,test_pred))
print('mean squared error:',metrics.mean_squared_error(y_test,test_pred))
rms=np.sqrt(metrics.mean_squared_error(y_test,test_pred))
print('root mean squared error:',rms)
```

```
mean absolute error: 0.4450284959258517
mean squared error: 0.3146880803682988
root mean squared error: 0.5609706590975136
```

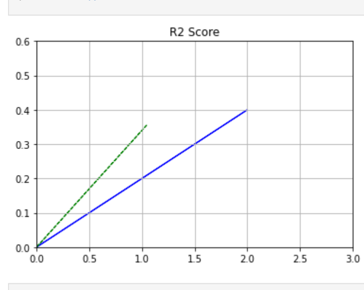
In [56]:
```python
from sklearn.metrics import r2_score
r2_score(y_test,test_pred)
```

Out[56]:
```
0.3896480246485313
```

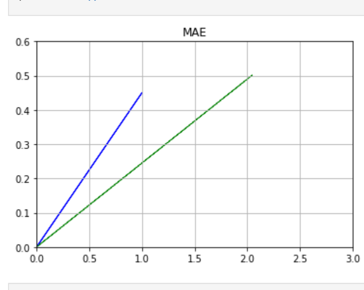
In [66]:
```python
import matplotlib.pyplot as plt1
ax=plt1.axes()
color1="green"
color2='blue'
ax.arrow(0,0,1,0.56, head_width=0.00,head_length=0,fc=color2,ec=color2)
ax.arrow(0,0,2,0.63, head_width=0.00,head_length=0.05,fc=color1,ec=color1,linestyle="--")
ax.set_ylim([0,0.8])
ax.set_xlim([0,4])
plt.grid()
plt.title('RMSE_score')
plt1.show()
```



In [67]:
```python
import matplotlib.pyplot as plt1
ax=plt1.axes()
color1= 'green'
color2= 'blue'
ax.arrow (0,0,2,0.40, head_width=0.00, head_length=0,fc=color2, ec=color2)
ax.arrow(0,0,1,0.34,head_width=0.00,head_length=0.05, fc=color1, ec=color1, linestyle="--")
ax.set_ylim([0,0.6])
ax.set_xlim([0,3])
plt.grid()
plt.title("R2 Score")
plt1.show()
```



In [69]:
```python
import matplotlib.pyplot as plt1
ax=plt1.axes()
color1= 'green'
color2= 'blue'
ax.arrow(0,0,1,0.45, head_width=0.00, head_length=0, fc=color2, ec=color2)
ax.arrow(0,0,2,0.49, head_width=0.00, head_length=0.05, fc=color1, ec=color1, linestyle="--")
ax.set_ylim([0,0.6])
ax.set_xlim([0,3])
plt.grid()
plt.title('MAE')
plt1.show()
```



In [ ]: