In [2]:

```
# import pyhton libraries
import numpy as np # It will tske care of numerical data
import pandas as pd # It will import excel file
# import data visualization library
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

(https://getlin

In [11]:

```
data=pd.read_csv(r"D:\PYTHON PROGRAMMES\WineQT.csv")
data
```

Out[11]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
...	...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71

1143 rows × 12 columns

In [5]:

```
# Check rows and columns in the data set using .shape
data.shape
```

Out[5]:

```
(1143, 13)
```

In [6]:

```
# Checking information about the dataset using .info()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

(https://getlin

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
Id                      0
dtype: int64
```

In [8]:

```
data.describe()
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total s di
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.0(
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45.91
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32.78
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.0(
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21.0(
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37.0(
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61.0(
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289.0(

(https://getlin

In [20]:

```
X=data[['fixed acidity','volatile acidity','residual sugar','chlorides','total sulfur di
```

In [13]:

```
y=data[['quality']]
```

In [14]:

```
X
```

Out[14]:

	fixed acidity	volatile acidity	residual sugar	chlorides	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	1.9	0.076	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	2.6	0.098	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	2.3	0.092	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	1.9	0.075	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	1.9	0.076	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	2.3	0.076	40.0	0.99574	3.42	0.75	11.0
1139	6.8	0.620	1.9	0.068	38.0	0.99651	3.42	0.82	9.5
1140	6.2	0.600	2.0	0.090	44.0	0.99490	3.45	0.58	10.5
1141	5.9	0.550	2.2	0.062	51.0	0.99512	3.52	0.76	11.2
1142	5.9	0.645	2.0	0.075	44.0	0.99547	3.57	0.71	10.2

(https://getlin

1143 rows × 9 columns

In [15]:

```
y
```

Out[15]:

	quality
0	5
1	5
2	5
3	6
4	5
...	...
1138	6
1139	6
1140	5
1141	6
1142	5

1143 rows × 1 columns

In [16]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=4
```

In [17]:

```
 from sklearn.linear_model import LinearRegression
```

In [18]:

```
reg = LinearRegression().fit(X_train, y_train)
reg.score(X_test,y_test)
```

(https://getlin

Out[18]:

0.3424875144168652

MODEL ACCURACY IS 34.24%

BY OLS SUMMARY

In [21]:

```
import statsmodels.api as sm
model = sm.OLS(y,X)
results = model.fit()
print(results.summary())
```

(https://getlin

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                quality   R-squared:
0.373
Model:                            OLS   Adj. R-squared:
0.368
Method:                 Least Squares   F-statistic:                     7
4.99
Date:                Sun, 03 Sep 2023   Prob (F-statistic):          1.29e
-108
Time:                        21:38:16   Log-Likelihood:                -11
07.5
No. Observations:                1143   AIC:                             2
235.
Df Residuals:                    1133   BIC:                             2
285.
Df Model:                           9
Covariance Type:            nonrobust
==========================================================================
==============
                           coef    std err          t      P>|t|      [0.0
25      0.975]
--------------------------------------------------------------------------
--------------
fixed acidity            0.0168      0.029      0.587      0.557      -0.0
39       0.073
volatile acidity        -1.0828      0.119     -9.106      0.000      -1.3
16      -0.849
residual sugar           0.0152      0.018      0.828      0.408      -0.0
21       0.051
chlorides               -1.8085      0.473     -3.822      0.000      -2.7
37      -0.880
total sulfur dioxide    -0.0024      0.001     -3.874      0.000      -0.0
04      -0.001
density                -20.1290     25.189     -0.799      0.424     -69.5
51      29.293
pH                      -0.3703      0.221     -1.677      0.094      -0.8
04       0.063
sulphates                0.8764      0.133      6.571      0.000       0.6
15       1.138
alcohol                  0.2748      0.031      8.941      0.000       0.2
14       0.335
const                   24.1646     24.680      0.979      0.328     -24.2
59      72.588
==========================================================================
====
Omnibus:                       19.311   Durbin-Watson:
1.779
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               2
9.120
Skew:                          -0.152   Prob(JB):                     4.75
e-07
Kurtosis:                       3.721   Cond. No.                     1.07
e+05
==========================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

(https://getlin

```
[2] The condition number is large, 1.07e+05. This might indicate that ther
e are
strong multicollinearity or other numerical problems.
```

BY RIDGE ALGORITHM

In [22]:

```
# Using Ridge Algorithm
from sklearn.linear_model import Ridge
```

In [23]:

(https://getlin

```
clf = Ridge(alpha=1.0)
clf.fit(X_train, y_train)
clf.score(X_train, y_train)
```

Out[23]:

```
0.37752215899941277
```

In [24]:

```
clf.score(X_test, y_test)
```

Out[24]:

```
0.3469752193868091
```

MODEL PREDICTED GRAPH

In [26]:

```
y_pred = reg.predict(X_test)
y_pred
```

```
       [5.97571082],
       [5.11257628],
       [5.52110827],
       [6.14184198],
       [5.43275836],
       [6.61780163],
       [5.75538717],
       [5.56420156],
       [5.20533374],
       [5.46673522],
       [6.03848713],
       [6.83833721],
       [6.51449947],
       [6.42228557],
       [5.39653428],
       [6.90906969],
       [5.30737726],
       [5.01781887],
       [5.03061961],
       [5.59682965],
```

In [27]:

```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X,y)
ExtraTreesRegressor()
```

Out[27]:

```
▾ ExtraTreesRegressor
ExtraTreesRegressor()
```

(https://getlin

In [28]:

```
print(model.feature_importances_)
```

```
[0.07082678 0.1656245  0.06456725 0.06254629 0.08856444 0.06467975
 0.07386694 0.13538326 0.27394079 0.        ]
```

In [29]:

```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```