In [7]:
```python
# import libraries
import pandas as pd
import numpy as np
```

In [8]:
```python
# load the data
df = pd.read_csv(r"C:\Users\linht\Downloads\PortfolioProjects\Project3\wine.csv")
df.head(10)
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	3.8	0.310	0.02	11.10	0.036	20	114	0.99248	3.75	0.44	
1	3.9	0.225	0.40	4.20	0.030	29	118	0.98900	3.57	0.36	
2	4.2	0.170	0.36	1.80	0.029	93	161	0.98999	3.65	0.89	
3	4.2	0.215	0.23	5.10	0.041	64	157	0.99688	3.42	0.44	
4	4.4	0.320	0.39	4.30	0.030	31	127	0.98904	3.46	0.36	
5	4.4	0.460	0.10	2.80	0.024	31	111	0.98816	3.48	0.34	
6	4.4	0.540	0.09	5.10	0.038	52	97	0.99022	3.41	0.40	
7	4.5	0.190	0.21	0.95	0.033	89	159	0.99332	3.34	0.42	
8	4.6	0.445	0.00	1.40	0.053	11	178	0.99426	3.79	0.55	
9	4.7	0.145	0.29	1.00	0.042	35	90	0.99080	3.76	0.49	

In [9]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3961 entries, 0 to 3960
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         3961 non-null   float64
 1   volatile acidity      3961 non-null   float64
 2   citric acid           3961 non-null   float64
 3   residual sugar        3961 non-null   float64
 4   chlorides             3961 non-null   float64
 5   free sulfur dioxide   3961 non-null   int64  
 6   total sulfur dioxide  3961 non-null   int64  
 7   density               3961 non-null   float64
 8   pH                    3961 non-null   float64
 9   sulphates             3961 non-null   float64
 10  alcohol               3961 non-null   float64
 11  quality               3961 non-null   int64  
dtypes: float64(9), int64(3)
memory usage: 371.5 KB
```

```
In [10]: df.describe(include = "all")
```

Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	tota
count	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000	3961
mean	6.839346	0.280538	0.334332	5.914819	0.045905	34.894471	137
std	0.866860	0.103437	0.122446	4.861646	0.023103	17.217121	43
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9
25%	6.300000	0.210000	0.270000	1.600000	0.035000	23.000000	106
50%	6.800000	0.260000	0.320000	4.700000	0.042000	33.000000	133
75%	7.300000	0.330000	0.390000	8.900000	0.050000	45.000000	166
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440

```
In [11]: df.duplicated().sum()
```

```
Out[11]: 0
```

```
In [12]: # finding the correlation between features to the wine quality
for param in df.drop('quality', axis = 1).columns:
    print(f"Correlation of quality and {param} is ", df[[param, 'quality']].corr())
```

```
Correlation of quality and fixed acidity is                  fixed acidity   quality
fixed acidity       1.000000 -0.124636
quality            -0.124636  1.000000
Correlation of quality and volatile acidity is                    volatile acidity
quality
volatile acidity          1.000000 -0.190678
quality                  -0.190678  1.000000
Correlation of quality and citric acid is               citric acid   quality
citric acid     1.000000  0.007065
quality         0.007065  1.000000
Correlation of quality and residual sugar is                  residual sugar   quali
ty
residual sugar        1.000000 -0.117339
quality              -0.117339  1.000000
Correlation of quality and chlorides is             chlorides   quality
chlorides   1.000000 -0.217739
quality    -0.217739  1.000000
Correlation of quality and free sulfur dioxide is                       free sulfur
dioxide  quality
free sulfur dioxide              1.00000  0.01038
quality                          0.01038  1.00000
Correlation of quality and total sulfur dioxide is                        total sulf
ur dioxide   quality
total sulfur dioxide              1.000000 -0.183352
quality                          -0.183352  1.000000
Correlation of quality and density is            density   quality
density  1.000000 -0.337805
quality -0.337805  1.000000
Correlation of quality and pH is                 pH   quality
pH       1.000000  0.123829
quality  0.123829  1.000000
Correlation of quality and sulphates is             sulphates  quality
sulphates     1.0000   0.0532
quality       0.0532   1.0000
Correlation of quality and alcohol is            alcohol   quality
alcohol  1.000000  0.462869
quality  0.462869  1.000000
```

In [13]:
```
# import libraries
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
%matplotlib inline
```

In [14]:
```
# create the linear regression object
lm = LinearRegression()
lm
```

Out[14]:

▾ LinearRegression

LinearRegression()

In [15]:
```
# fit the linear model using 'alcohol' feature
lm.fit(df[['alcohol']], df[['quality']])
```

Out[15]:

```
▾ LinearRegression
LinearRegression()
```

In [16]:

```python
# ouput the prediction
y_hat = lm.predict(df[['alcohol']])
y_hat[0:5]
```

Out[16]:

```
array([[6.46816746],
       [6.60366256],
       [6.33267236],
       [4.97772138],
       [6.60366256]])
```

In [17]:

```python
# value of the intercept
lm.intercept_
```

Out[17]:

```
array([2.26781941])
```

In [18]:

```python
# value of the slope
lm.coef_
```

Out[18]:

```
array([[0.33873775]])
```

In [19]:

```python
# develop a model using the predictor variables
Z = df[['density', 'chlorides', 'volatile acidity']]
```

In [20]:

```python
# fit the linear model
lm.fit(Z, df['quality'])
```

Out[20]:

```
▾ LinearRegression
LinearRegression()
```

In [21]:

```python
# value of the intercept
lm.intercept_
```

Out[21]:

```
96.47901953423002
```

In [22]:

```python
# values of the coefficients
lm.coef_
```

Out[22]:

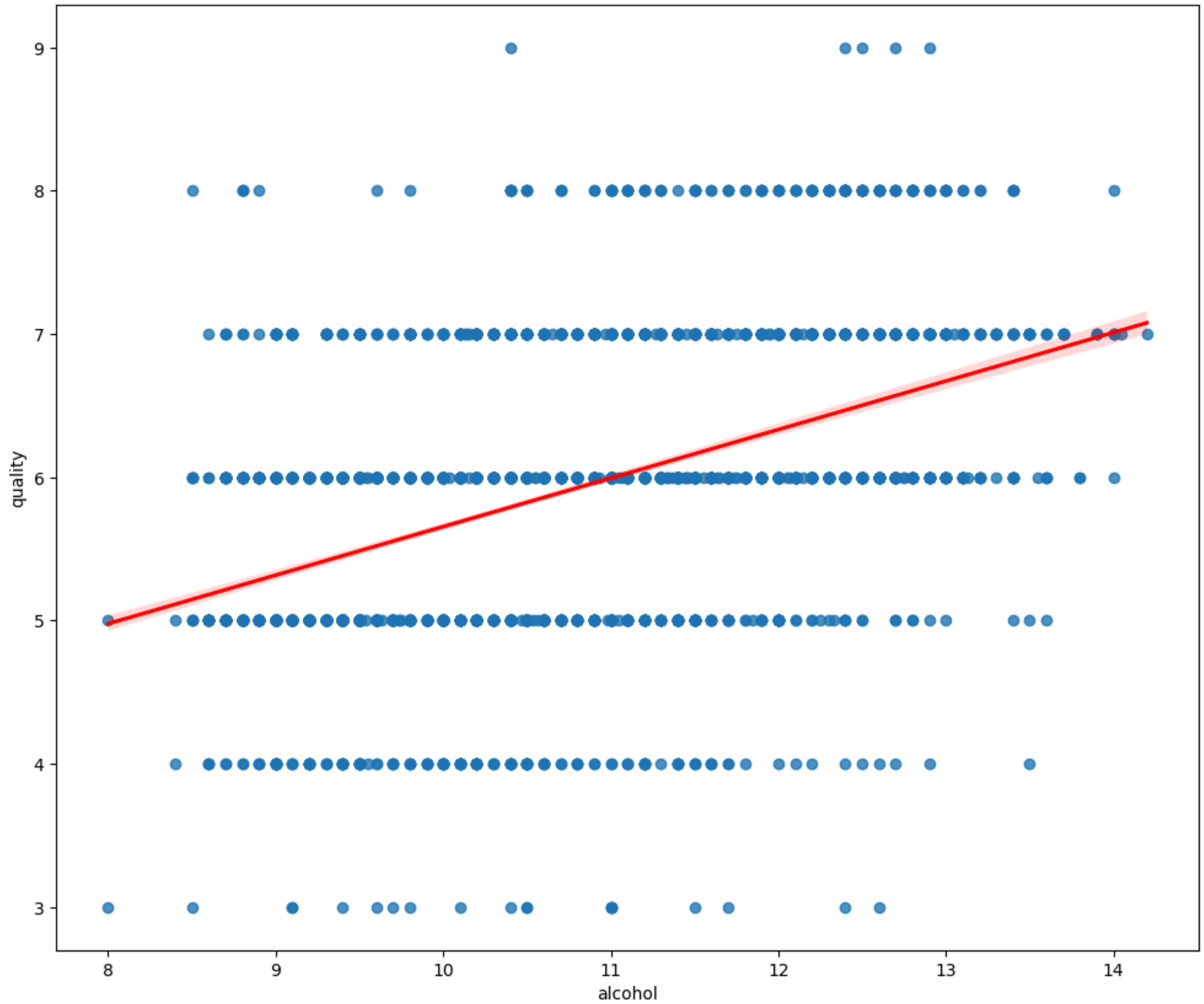
```
array([-90.56779731,  -4.97496064,  -1.39189898])
```

In [23]:

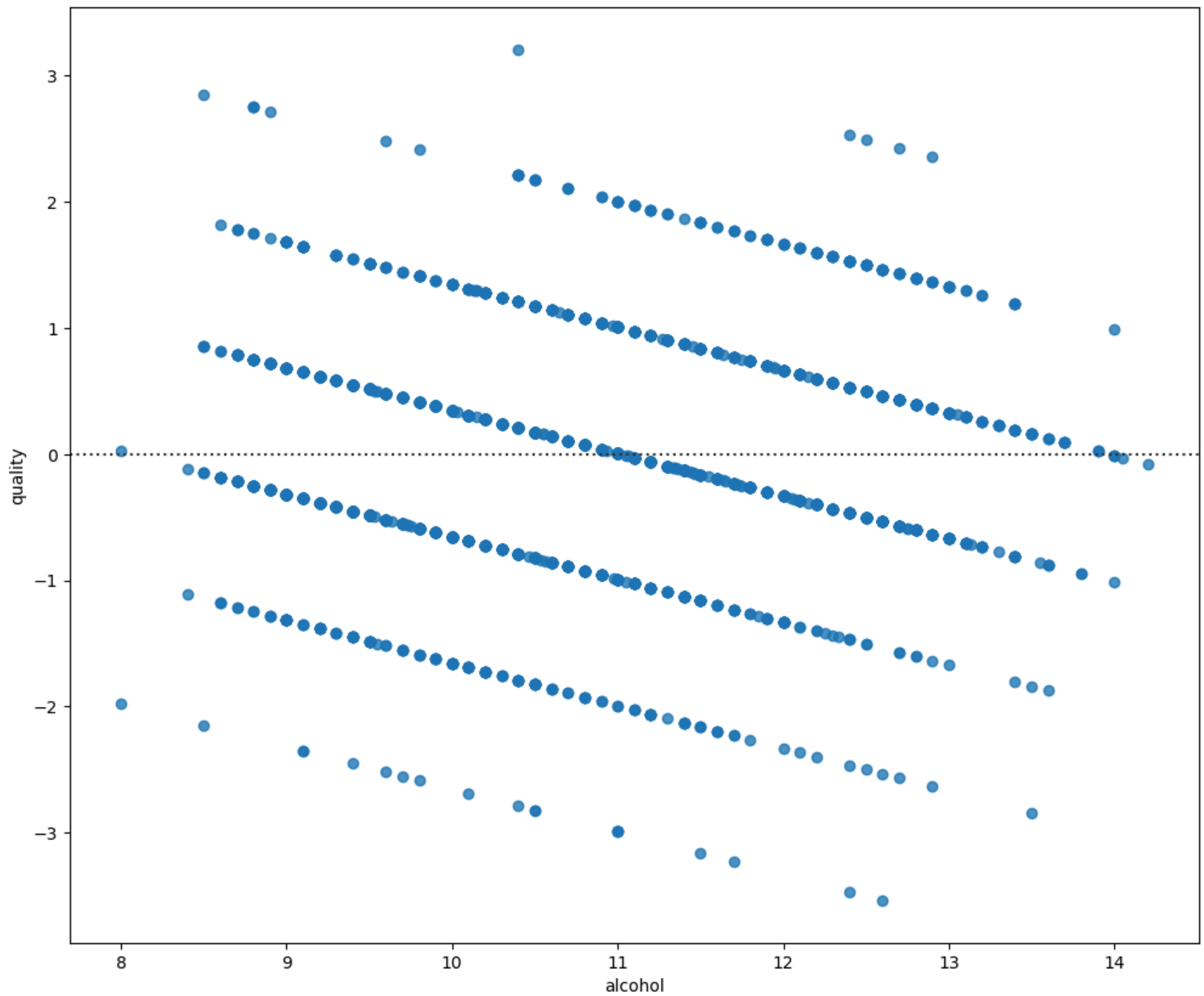
```python
# visualize regression plot 'alcohol' as potential predictor variable of 'quality'
width = 12
height = 10
plt.figure(figsize = (width, height))
sns.regplot(x = "alcohol", y = "quality", line_kws = {'color': 'red'}, data = df)
```

Out[23]:

```
<Axes: xlabel='alcohol', ylabel='quality'>
```

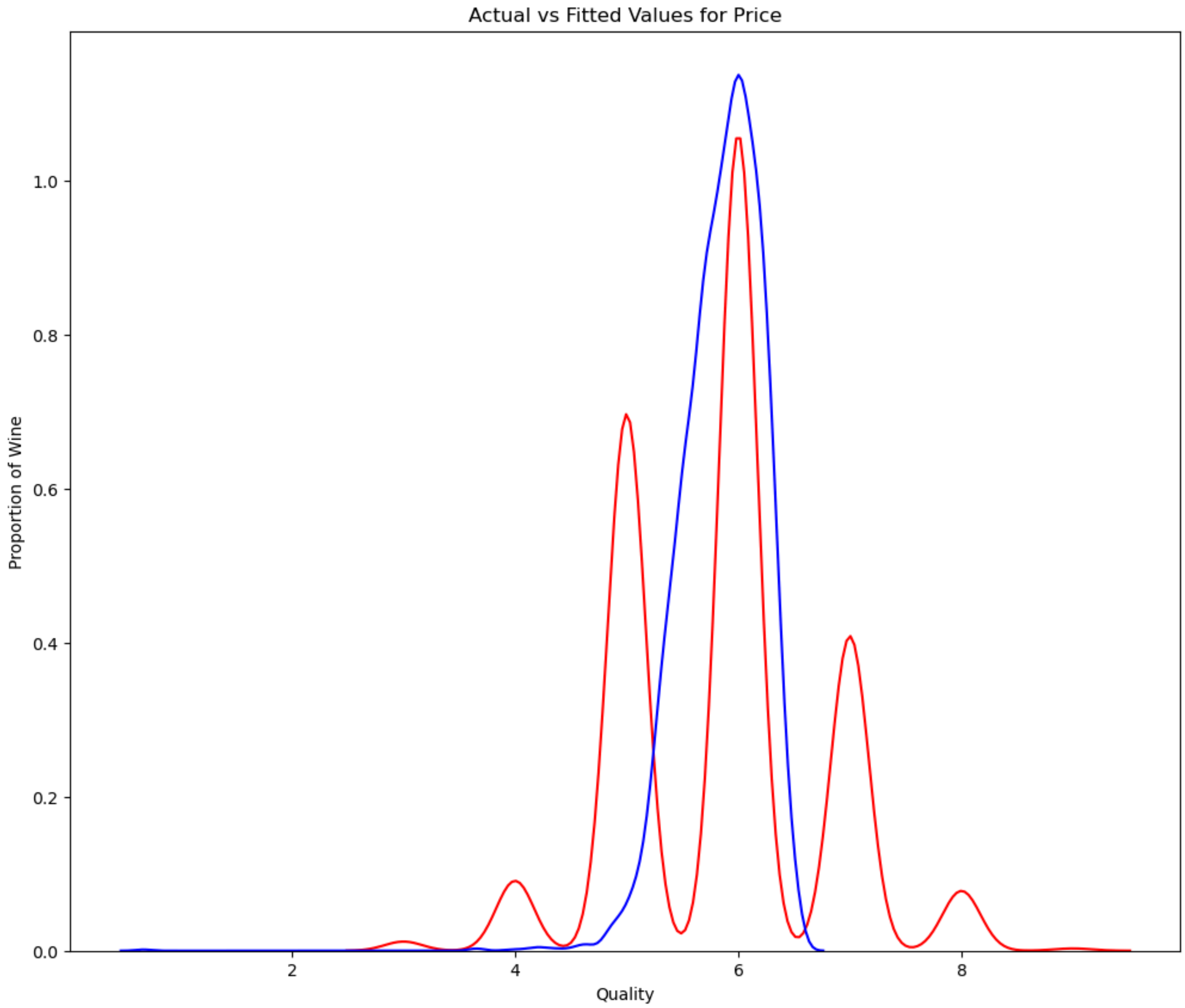


In [24]:
```
# visualize residual plot 'alcohol' as potential predictor variable of 'quality'
plt.figure(figsize = (width, height))
sns.residplot(x = df['alcohol'], y = df['quality'])
plt.show()
```



In [25]:

```
# distribution plot
import warnings
warnings.filterwarnings('ignore')
yhat = lm.predict(Z)
plt.figure(figsize = (width, height))
ax1 = sns.distplot(df['quality'], hist = False, color = "r", label = "Actual Values
sns.distplot(yhat, hist = False, color = "b", label = "Fitted Values", ax = ax1)
plt.title("Actual vs Fitted Values for Price")
plt.xlabel("Quality")
plt.ylabel("Proportion of Wine")
plt.show()
plt.close()
```

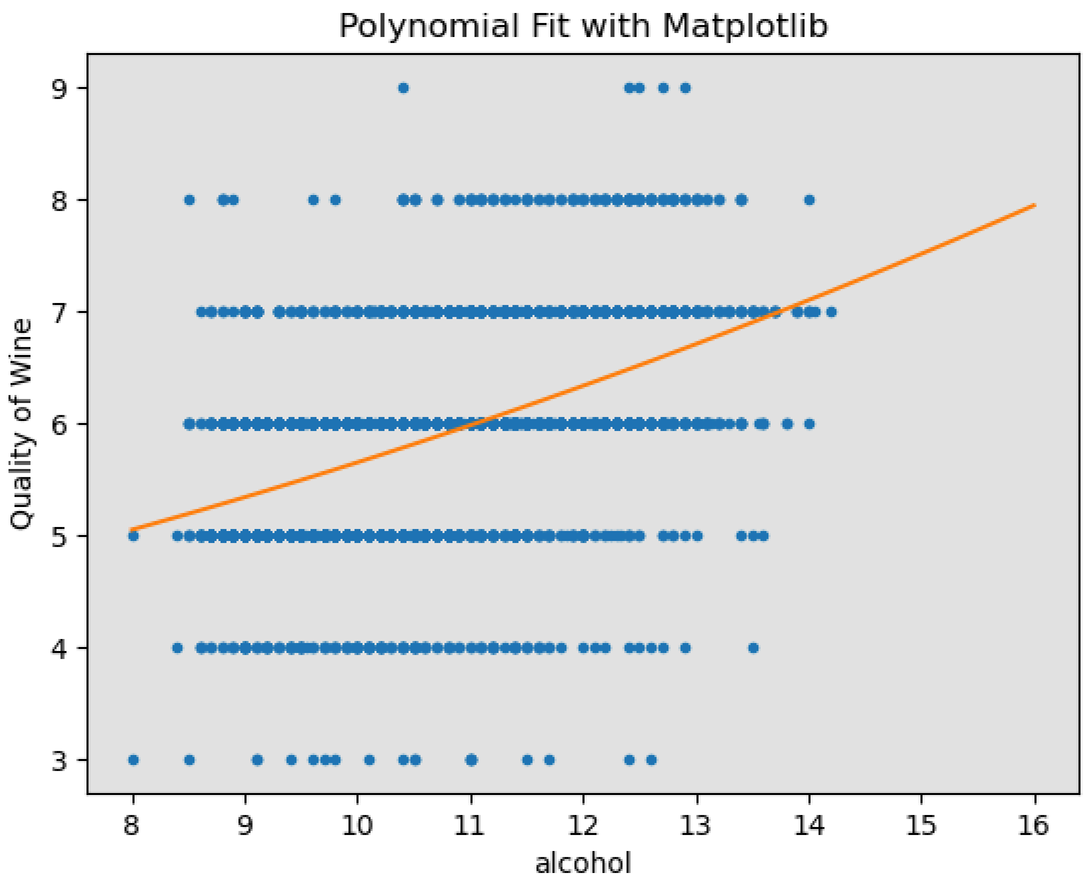


In [26]:
```
# plot polynomial regression
def PlotPolly(model, independent_variable, dependent_variable, Name):
    x_new = np.linspace(8,16)
    y_new = model(x_new)
    plt.plot(independent_variable, dependent_variable, '.', x_new, y_new, '-')
    plt.title("Polynomial Fit with Matplotlib")
    ax = plt.gca()
    ax.set_facecolor((0.898,0.898,0.898))
    fig = plt.gcf()
    plt.xlabel(Name)
    plt.ylabel("Quality of Wine")
    plt.show()
    plt.close()
```

In [27]:
```
# get the variables
x = df['alcohol']
y = df['quality']
# use the polynomial of the 3rd order
f = np.polyfit(x, y, 2)
p = np.poly1d(f)
print(p)
```

```
        2
0.0102 x + 0.1171 x + 3.456
```

In [28]:
```python
# plot the function
PlotPolly(p,x,y,'alcohol')
np.polyfit(x,y,2)
```



Out[28]:
```
array([0.01020278, 0.11706229, 3.45602745])
```

In [29]:
```python
# perform a polynomial transform on multiple features
from sklearn.preprocessing import PolynomialFeatures
pr = PolynomialFeatures()
Z_pr = pr.fit_transform(Z)
Z.shape
Z_pr.shape
```

Out[29]:
```
(3961, 10)
```

In [30]:
```python
# pipeline
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
Input = [('scale', StandardScaler()), ('polynomial', PolynomialFeatures(include_bia
pipe = Pipeline(Input)
pipe.fit(Z,y)
ypipe = pipe.predict(Z)
ypipe[0:5]
```

Out[30]:
```
array([5.9797678 , 6.58972241, 6.47370237, 5.72986272, 6.5696553 ])
```

In [31]:
```python
# in-sample evaluation
# simple linear regression
```

```
lm.fit(df[['alcohol']], df[['quality']])
print("The R-squared is: ", lm.score(df[['alcohol']], df[['quality']]))
```

```
The R-squared is:  0.21424800749926098
```

In [32]:
```
y_hat = lm.predict(df[['alcohol']])
print("The output of the first five predicted values is: ", y_hat[0:5])
```

```
The output of the first five predicted values is:  [[6.46816746]
 [6.60366256]
 [6.33267236]
 [4.97772138]
 [6.60366256]]
```

In [33]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(df['quality'], y_hat)
print("The mean square error of quality and predicted value is: ", mse)
```

```
The mean square error of quality and predicted value is:  0.623191969587994
```

In [34]:
```
# multiple linear regression
lm.fit(Z, df['quality'])
print("The R-squared is: ", lm.score(Z, df['quality']))
```

```
The R-squared is:  0.15868996194749496
```

In [35]:
```
Yhat = lm.predict(Z)
print("The mean square error of quality and predicted value using multifit is: ", m
```

```
The mean square error of quality and predicted value using multifit is:  0.667255908
5462297
```

In [36]:
```
# polynomial fit
from sklearn.metrics import r2_score
r_squared = r2_score(y, p(x))
print("The R-squared value is: ", r_squared)
```

```
The R-squared value is:  0.21456475059065194
```

In [37]:
```
print("The mean square error of quality and predicted value using polyfit is: ", me
```

```
The mean square error of quality and predicted value using polyfit is:  0.6229407557
78979
```