In [ ]:

Collect the data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
sal = pd.read_csv('SalaryData.csv')
```

In [3]:
```
sal
```

Out[3]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

YearsExperience = feature, Salary = target

Process the data

Let the regression line be:

y = b0 + b1.X

y = b0*1 + b1.X

y = [b0 b1].[1 X]^T

In [6]:
```
pip install statsmodels
```

```
Requirement already satisfied: statsmodels in c:\users\urvi sharma\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: numpy>=1.15 in c:\users\urvi sharma\anaconda3\lib\site-packages (from statsmodels) (1.20.3)
Requirement already satisfied: scipy>=1.1 in c:\users\urvi sharma\anaconda3\lib\site-packages (from statsmodels) (1.7.1)
Requirement already satisfied: pandas>=0.21 in c:\users\urvi sharma\anaconda3\lib\site-packages (from statsmodels) (1.3.4)
Requirement already satisfied: patsy>=0.5 in c:\users\urvi sharma\anaconda3\lib\site-packages (from statsmodels) (0.5.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\urvi sharma\anaconda3\lib\site-packages (from pandas>=0.21->s
tatsmodels) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\urvi sharma\anaconda3\lib\site-packages (from pandas>=0.21->statsmodel
s) (2021.3)
Requirement already satisfied: six in c:\users\urvi sharma\anaconda3\lib\site-packages (from patsy>=0.5->statsmodels) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [7]:
```
# Adding constant 1 to each data point (in order to construct feature matrix of X) as a column vector

import statsmodels.api as sm
```

In [8]:
```
y = sal['Salary']
y
```

Out[8]:
```
0      39343
1      46205
2      37731
3      43525
4      39891
5      56642
6      60150
7      54445
8      64445
9      57189
10     63218
11     55794
12     56957
13     57081
14     61111
15     67938
16     66029
17     83088
18     81363
19     93940
20     91738
21     98273
22    101302
23    113812
24    109431
25    105582
26    116969
27    112635
28    122391
29    121872
Name: Salary, dtype: int64
```

In [9]:
```
X = sm.add_constant(sal['YearsExperience'])
X
```

```
C:\Users\Urvi Sharma\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a future version of pandas
all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[9]:

	const	YearsExperience
0	1.0	1.1
1	1.0	1.3
2	1.0	1.5
3	1.0	2.0
4	1.0	2.2
5	1.0	2.9
6	1.0	3.0
7	1.0	3.2
8	1.0	3.2
9	1.0	3.7
10	1.0	3.9
11	1.0	4.0
12	1.0	4.0
13	1.0	4.1
14	1.0	4.5
15	1.0	4.9
16	1.0	5.1
17	1.0	5.3
18	1.0	5.9
19	1.0	6.0
20	1.0	6.8
21	1.0	7.1
22	1.0	7.9
23	1.0	8.2
24	1.0	8.7
25	1.0	9.0
26	1.0	9.5
27	1.0	9.6
28	1.0	10.3
29	1.0	10.5

Divide the data into training and testing

In [11]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 10)
```

In [12]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[12]:
```
((24, 2), (6, 2), (24,), (6,))
```

In [13]:
```
X_train
```

Out[13]:

	const	YearsExperience
13	1.0	4.1
27	1.0	9.6
12	1.0	4.0
1	1.0	1.3
19	1.0	6.0
14	1.0	4.5
18	1.0	5.9
6	1.0	3.0
11	1.0	4.0
23	1.0	8.2
24	1.0	8.7
28	1.0	10.3
22	1.0	7.9
10	1.0	3.9
26	1.0	9.5
29	1.0	10.5
8	1.0	3.2
25	1.0	9.0
16	1.0	5.1
17	1.0	5.3
0	1.0	1.1
15	1.0	4.9
4	1.0	2.2
9	1.0	3.7

In [14]:
```
X_test
```

Out[14]:

	const	YearsExperience
20	1.0	6.8
7	1.0	3.2
5	1.0	2.9
2	1.0	1.5
3	1.0	2.0
21	1.0	7.1

In [15]:
```
y_train
```

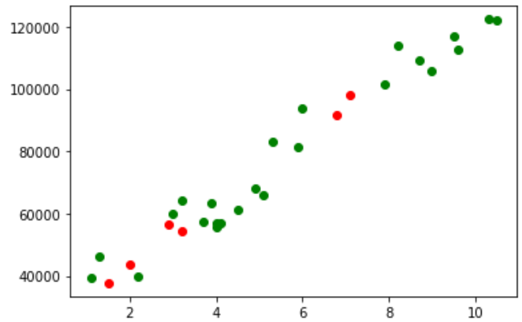
Out[15]:
```
13     57081
27    112635
12     56957
1      46205
19     93940
14     61111
18     81363
6      60150
11     55794
23    113812
24    109431
28    122391
22    101302
10     63218
26    116969
29    121872
8      64445
25    105582
16     66029
17     83088
0      39343
15     67938
4      39891
9      57189
Name: Salary, dtype: int64
```

In [16]:
```
y_test
```

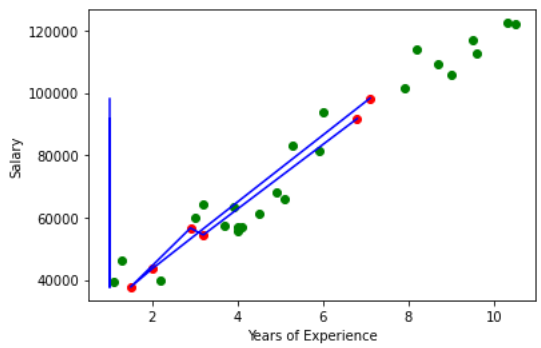
Out[16]:
```
20    91738
7     54445
5     56642
2     37731
3     43525
21    98273
Name: Salary, dtype: int64
```

Perform Data Exploration

In [18]:
```
plt.scatter(X_train['YearsExperience'],y_train, c = 'green' )
plt.scatter(X_test['YearsExperience'], y_test, c = 'red')
plt.show();
```



In [21]:
```
plt.scatter(X_train['YearsExperience'],y_train, c = 'green' )
plt.scatter(X_test['YearsExperience'], y_test, c = 'red')
plt.plot(X_test, y_test, c= 'blue')
plt.xlabel('Years of Experience');
plt.ylabel('Salary');
plt.show();
```



Model Building

In [22]:
```
sal_lr = sm.OLS(y_train, X_train)
```

Fitting the model

In [24]:
```
sal_lr = sal_lr.fit()
sal_lr
```

Out[24]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x2977d7898b0>
```

In [25]:
```
# Getting paramters

sal_lr.params
```

Out[25]:
```
const              26089.096632
YearsExperience     9356.862994
dtype: float64
```

Model Diagnosis

Co-efficient of Determination

Hypothesis test for the regression coefficient

Analysis of variance for overall model validity

Residual Analysis to validate the regression model assumptions

Outlier analysis, since the presence of outliers can significantly impact the regression parameter

In [26]:
```
sal_lr.summary2()
```

Out[26]:

Model:	OLS	Adj. R-squared:	0.947
Dependent Variable:	Salary	AIC:	490.1973
Date:	2022-10-03 11:11	BIC:	492.5534
No. Observations:	24	Log-Likelihood:	-243.10
Df Model:	1	F-statistic:	413.4
Df Residuals:	22	Prob (F-statistic):	9.45e-16
R-squared:	0.949	Scale:	4.0119e+07

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	26089.0966	2909.0925	8.9681	0.0000	20056.0080	32122.1852
YearsExperience	9356.8630	460.2197	20.3313	0.0000	8402.4258	10311.3002

Omnibus:	2.696	Durbin-Watson:	2.218
Prob(Omnibus):	0.260	Jarque-Bera (JB):	1.670
Skew:	0.402	Prob(JB):	0.434
Kurtosis:	1.989	Condition No.:	15

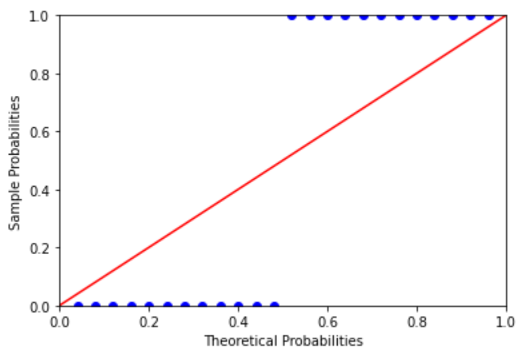
Checking for normal dist of residuals

In [27]:
```
## Calculating the residuals

sal_lr.resid
```

Out[27]:
```
13    -7371.234906
27    -3279.981370
12    -6559.548607
1      7951.981476
19    11709.725406
14    -7083.980103
18       68.411706
6      5990.314387
11    -7722.548607
23    10996.626821
24     1937.195324
28      -73.785466
22     1293.685719
10      637.137693
26     1989.704929
29    -2464.158065
8      8413.941788
25    -4718.863574
16    -7780.097899
17     7407.529502
0      2961.354075
15    -3999.725301
4     -6783.195218
9     -3520.489709
dtype: float64
```

In [28]:
```
# Usinf probability - probability plot
# Used to compare the cumulative dist of residual

probplot = sm.ProbPlot(sal_lr.resid)
probplot.ppplot(line = '45')
plt.show();
```

```
C:\Users\Urvi Sharma\anaconda3\lib\site-packages\statsmodels\graphics\gofplots.py:993: UserWarning: marker is redundantly defin
ed by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



Conclusion: The residuals got aren't normally distributed

Test of homoscedasticity

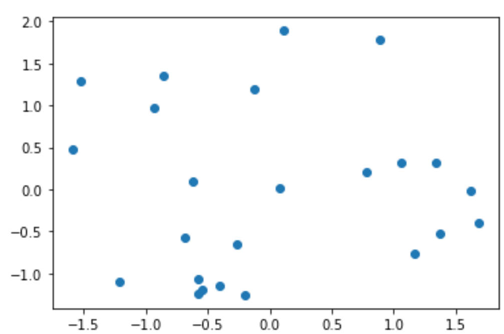
It means that residuals have constant variance.

If the data points in resdiual plot fo not have any pattern, it is satisfied.

It can be deducted from the residual plot

In [30]:
```
def standardisation(vals):
    return (vals-vals.mean())/vals.std()
```

In [31]:
```
# Plotting residual plot

plt.scatter(standardisation(sal_lr.fittedvalues), standardisation(sal_lr.resid))# fittedvalues = predicted values
plt.show();
```



Remarks: It shows that residuals have constant variance