SIMPLE LINEAR REGRESSION

Predicting the salary of MBA graduates based on there Experience

In [10]:

```
# Importing all required libraries for building the regression model

import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
```

In [11]:

```
# Load the dataset into dataframe

mba_salary_df=pd.read_csv('mba_salary.csv')
mba_salary_df.head()
```

Out[11]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [16]:

```
mba_salary_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

y=mx+c OLS API available in statsmodel.api estimates only the coefficient of X parameter. To estimate regression coefficient c , a constant term of 1 needs to be added as a separate column. As the value of the columns remains same across all samples, the parameter estimated for this feature or column will be the intercept form

In [13]:

```
# Add constant term of 1 to the  dataset

X=sm.add_constant(mba_salary_df['YearsExperience'])
X.head()
```

Out[13]:

	const	YearsExperience
0	1.0	1.1
1	1.0	1.3
2	1.0	1.5

	const	YearsExperience
3	1.0	2.0
4	1.0	2.2

In [5]:

```
Y=mba_salary_df['Salary']
Y.head()
```

Out[5]:

```
0    39343.0
1    46205.0
2    37731.0
3    43525.0
4    39891.0
Name: Salary, dtype: float64
```

The method takes a seed value in parameter named random_state, to fix which samples go to training and which one goes to test set

In [14]:

```
# Split dataset into train and test set into 80:20 respectively

train_X,test_X,train_Y,test_Y=train_test_split(X,Y,train_size=0.8,random_state=100)

# Fit the regression model
mba_salary_1m=sm.OLS(train_Y,train_X).fit()
```

In [31]:

```
# Print the model parameters

mba_salary_1m.params
```

Out[31]:

```
const              26819.065119
YearsExperience     9361.116390
dtype: float64
```

In [36]:

```
# Preddicting using the validation set

pred_Y=mba_salary_1m.predict(test_X)
from sklearn.metrics import r2_score, mean_squared_error
np.abs(r2_score(test_Y,pred_Y))
```

Out[36]:

```
0.9720725422361338
```

In [37]:

```
np.sqrt(mean_squared_error(test_Y,pred_Y))
```

Out[37]:

```
4947.434596804257
```

MODEL DIAGNOSTICS

In [17]:

```
mba_salary_1m.summary2()
```

Out[17]:

Model:	OLS	Adj. R-squared:	0.949
Dependent Variable:	Salary	AIC:	487.9089
Date:	2020-08-23 13:00	BIC:	490.2650
No. Observations:	24	Log-Likelihood:	-241.95
Df Model:	1	F-statistic:	430.7
Df Residuals:	22	Prob (F-statistic):	6.14e-16
R-squared:	0.951	Scale:	3.6470e+07

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	26819.0651	2651.4098	10.1150	0.0000	21320.3777	32317.7526
YearsExperience	9361.1164	451.0656	20.7533	0.0000	8425.6635	10296.5693

Omnibus:	1.950	Durbin-Watson:	1.771
Prob(Omnibus):	0.377	Jarque-Bera (JB):	1.237
Skew:	0.265	Prob(JB):	0.539
Kurtosis:	2.022	Condition No.:	13

In [18]:

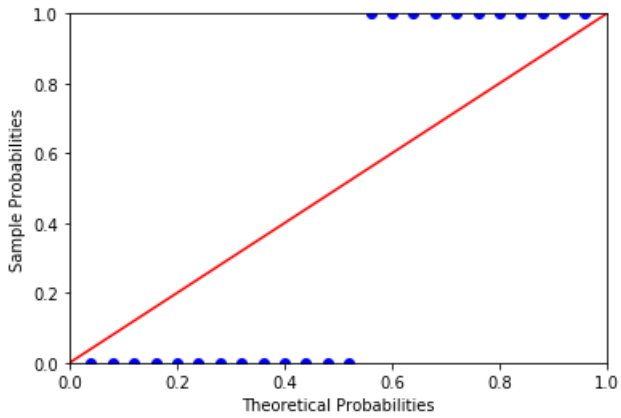
```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

Check for normal distribution of residual

In [25]:

```
mba_salary_resid=mba_salary_1m.resid
probplot=sm.ProbPlot(mba_salary_resid)
plt.figure(figsize=(8,6))
probplot.ppplot(line='45')
plt.show()
```

```
<Figure size 576x432 with 0 Axes>
```



The diagonal line is cumulative dsitribution of a normal distribution, whereas the dots represent the cumulative distribution of the residuals.

Test of Homoscedasticity

In [40]:

```
def get_standardized_values(vals):
```

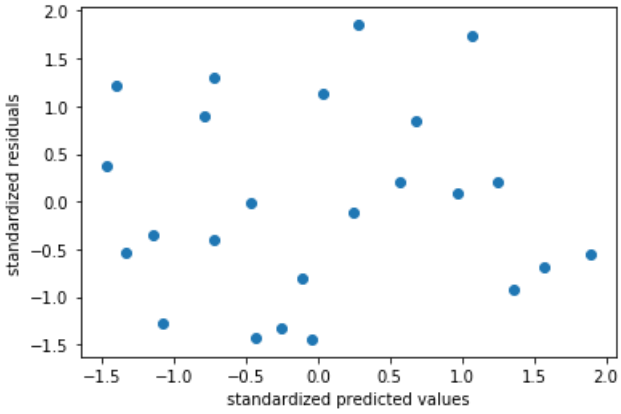
```
    return (vals-vals.mean())/vals.std()
```

In [43]:

```
plt.scatter(get_standardized_values(mba_salary_1m.fittedvalues),get_standardized_values(mba_salary_
resid))
plt.xlabel("standardized predicted values")
plt.ylabel("standardized residuals")
```

Out[43]:

```
Text(0, 0.5, 'standardized residuals')
```



The residuals are random and have no funnel shape, which means the residuals have constant variance

Outlier Analysis

In [44]:

```
from scipy.stats import zscore
```

In [45]:

```
mba_salary_df['z_score_salary']=zscore(mba_salary_df.Salary)
```

Any observation with a Z-score of more than 3 may be flagged as an outlier

In [46]:

```
mba_salary_df[(mba_salary_df.z_score_salary>3.0)|(mba_salary_df.z_score_salary<-3.0)]
```

Out[46]:

	YearsExperience	Salary	z_score_salary

So, there are no observations that are outliers as per the Z-Score