21. Linear Regression (Practical)

In [21]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [11]:
```
dataset = pd.read_csv(r'Data/placement.csv')
dataset.head(3)
```

Out[11]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25

In [12]:
```
dataset.isnull().sum()
```

Out[12]:
```
cgpa       0
package    0
dtype: int64
```

- data has to be in multidimensional or 2 dimentional at least

In [13]:
```
x = dataset["cgpa"]
x.ndim
```

Out[13]:
```
1
```

- So we will convert this data into 2 dimensional data:

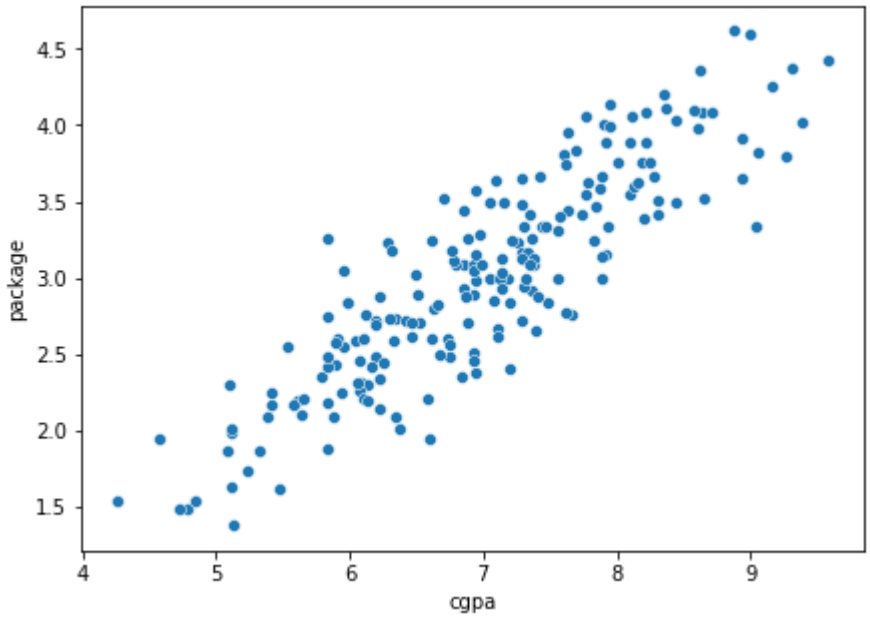
In [14]:
```
x = dataset[["cgpa"]]
x.ndim
```

Out[14]:
```
2
```

In [15]:
```
y = dataset['package']
```

- Before applying linear regression, check that if your data is following linearity or not

In [20]:
```
plt.figure(figsize=(7,5))
sns.scatterplot(x='cgpa', y='package', data=dataset)
plt.show()
```



- You can see the data is following simple linearity

In [22]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state
```

In [26]:
```
# y = mx + c
from sklearn.linear_model import LinearRegression
```

In [27]:
```
lr = LinearRegression()
# fit will train the data to fit linear equation,
# y = mx + c, this will search for best m and c value to train the data on this lin
lr.fit(x_train, y_train)
```

Out[27]:
```
▾ LinearRegression

LinearRegression()
```

- Now our model is trained now, and ready for testing

In [30]:
```
lr.predict([[6.89]])
```

```
C:\Users\rashi\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\bas
e.py:450: UserWarning: X does not have valid feature names, but LinearRegression was
fitted with feature names
  warnings.warn(
```

Out[30]:
```
array([2.92962016])
```

In [31]:
```
dataset.head(3)
```

Out[31]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25

- To check if prediction is model is good or now, we will use **accuracy score**

In [33]:
```
lr.score(x_test, y_test)*100
```

Out[33]:
```
77.30984312051673
```

- To improve accuracy we will change random_state value in following code and see if the model accuracy has increased:
- x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=42)

To find the equation manually

In [41]:
```
# y = mx + c
```

In [36]:
```
m = lr.coef_
m
```

Out[36]:
```
array([0.57425647])
```

In [37]:
```
c = lr.intercept_
c
```

Out[37]:
```
-1.0270069374542108
```

In [40]:
```
y = (m * 6.89) + c
y
```

Out[40]:
```
array([2.92962016])
```

--

To draw prediction line

In [46]:
```
# y_pred = lr.predict([['cgpa']]) = y_pred = lr.predict(x)
y_pred = lr.predict(x)
```

In [55]:
```
plt.figure(figsize=(7,5))
sns.scatterplot(x='cgpa', y='package', data=dataset)
# plt.plot(x,y)
plt.plot(dataset['cgpa'], y_pred, c='red')
plt.legend(["Original data", "Predict line"])
```

```
plt.savefig(r"Generated_images/predict.jpg")
plt.show()
```