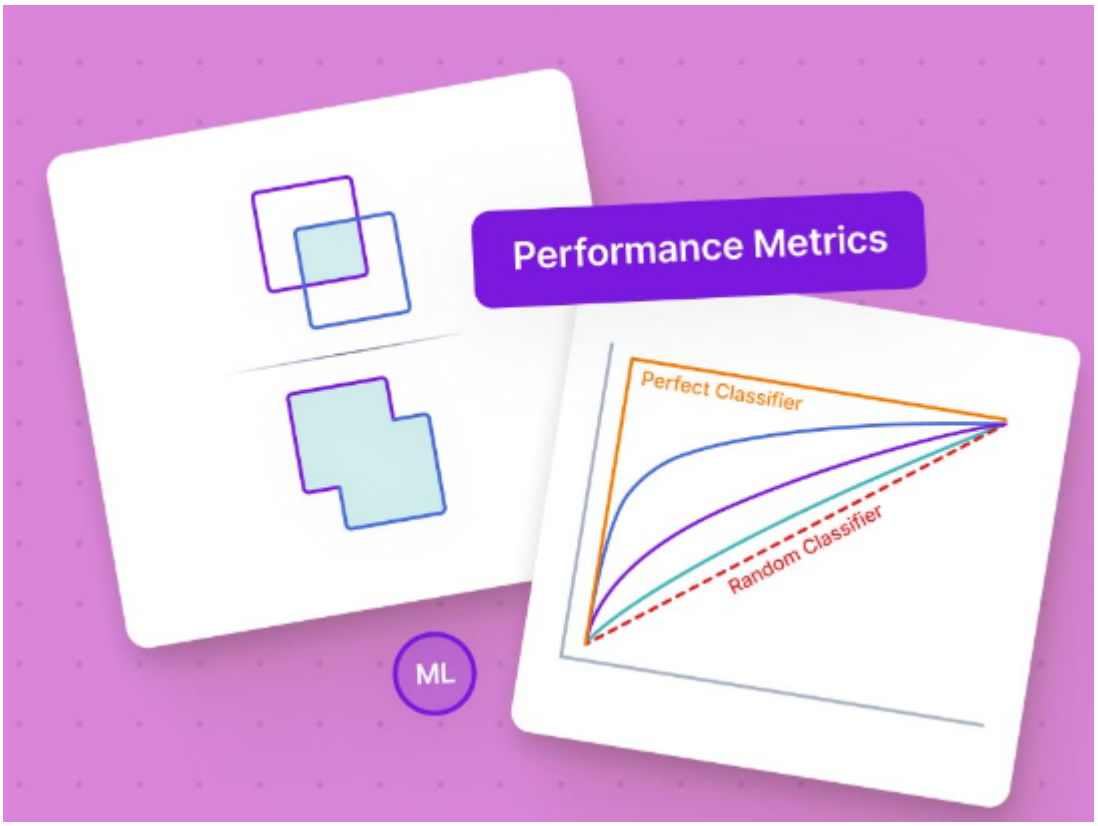
Regression Metrics

Regression metrics are used to evaluate the performance of a regression model. They measure the difference between the predicted and actual values, and can be used to compare different models or to track the performance of a model over time.

- We cannot calculate accuracy for a regression model.
- The skill or performance of a regression model must be reported as an error in those predictions.
- This makes sense if you think about it. If you are predicting a numeric value like a height or a dollar amount, you don't want to know if the model predicted the value exactly (this might be intractably difficult in practice); instead, we want to know how close the predictions were to the expected values.



1. **MAE**(Mean Absolute Error)
2. **MSE**(Mean Squared Error)
3. **RMSE** (Root Mean Squared Error)
4. **R2** (R squared)
5. **Adj R2** (Adjusted R squared)

Here are some detailed about regression metrics:

- **Mean Absolute Error (MAE)**: This metric measures the average of the absolute differences between the predicted and actual values. It gives equal weight to all errors and is less sensitive to outliers.
- **Mean Squared Error (MSE)**: This metric measures the average of the squared differences between the predicted and actual values. It is more sensitive to outliers than MAE.
- **Root Mean Squared Error (RMSE)**: This metric is the square root of MSE. It is in the same units as the response variable, making it easier to interpret.
- **R-squared (R2)**: This metric measures the proportion of variance in the target variable explained by the model. It ranges from 0 to 1, with higher values indicating better performance.
- **Adjusted R-squared (Adj R2)**: This metric is similar to R2 but accounts for the number of predictors used in the model. It penalizes overfitting by adjusting for the number of predictors.

Which metric to use?

The choice of metric depends on the specific problem and the desired outcome.

- **MAE** : is a good choice when the goal is to **minimize the overall error** in the model while avoiding large errors.
- **RMSE**: is a good choice when the goal is to **measure the absolute fit of the model**.
- **R-squared**: is a good choice when the goal is to **explain the variability in the target variable using the predictors**.
- **Adj R2** : is a good choice when the goal is to select a **parsimonious model that explains the variability in the target variable.**

Here is a table summarizing the different metrics:

Metric	Description	Units	Interpretation
MAE	Mean Absolute Error	Same as the response variable	Average of the absolute differences between the predicted and actual values
MSE	Mean Squared Error	Same as the response variable squared	Average of the squared differences between the predicted and actual values
RMSE	Root Mean Squared Error	Same as the response variable	Square root of the average of the squared differences between the predicted and actual values
R-squared	Coefficient of Determination	0 to 1	Proportion of variance in the target variable explained by the model

Metric Description Units Interpretation

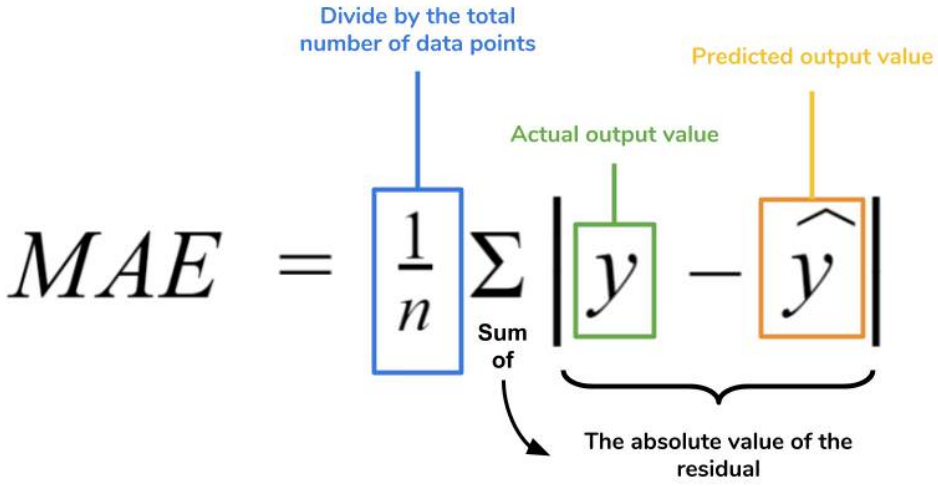
1.MAE (Mean Absolute Error)

Mean Absolute Error (MAE) is a regression metric that measures the average of the absolute differences between the predicted and actual values. It is a **robust** metric, meaning that it is not as sensitive to outliers as other metrics, such as Mean Squared Error (MSE).

Formula:

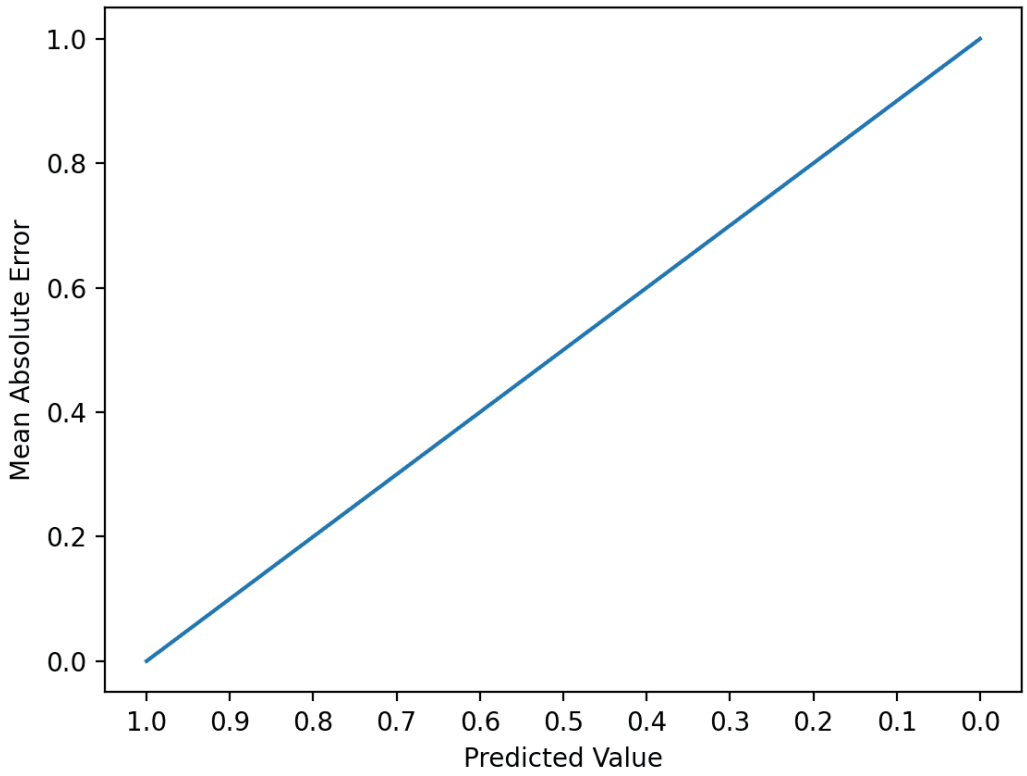
```
MAE = 1/n * Σ|y_true - y_pred|
```

- **y_true:** The actual value of the target variable.
- **y_pred:** The predicted value of the target variable.
- **n:** The number of observations.



Interpretation:

The MAE is the average distance between the predicted and actual values. **A lower MAE indicates that the model is closer to the actual values**, while a higher MAE indicates that the model is further away from the actual values.



Here are some of the advantages and disadvantages of MAE:

Advantages:

- **Robust to outliers:** MAE is not as sensitive to outliers as other metrics, such as MSE. This means that it is less likely to be affected by a few extreme values in the data.
- **Easy to interpret:** MAE is easy to understand and interpret. It is simply the average of the absolute differences between the predicted and actual values.
- **Scale-independent:** MAE is scale-independent, which means that it is not affected by the units of the data. This makes it a good choice for comparing models that are trained on different datasets with different scales.

Disadvantages:

- **Not as sensitive to large errors:** MAE is not as sensitive to large errors as other metrics, such as MSE. This means that it may not be as good at detecting models that are consistently over- or under-estimating the target variable.
- **Not differentiable:** MAE is not differentiable, which means that it cannot be used with gradient descent optimization algorithms.

Overall, MAE is a good choice for evaluating the performance of a regression model when the goal is to minimize the overall error in the model while avoiding large errors. However, it is important to be aware of the limitations of MAE, such as its lack of sensitivity to large errors.

Here is a table summarizing the advantages and disadvantages of MAE:

Advantage	Disadvantage
Robust to outliers	Not as sensitive to large errors
Easy to interpret	Not differentiable
Scale-independent	

Main goal of MAE is Reduce the error - outcome should be Low or small , & if you have more outliers then choose MAE

2.MSE (Mean Squared Error)

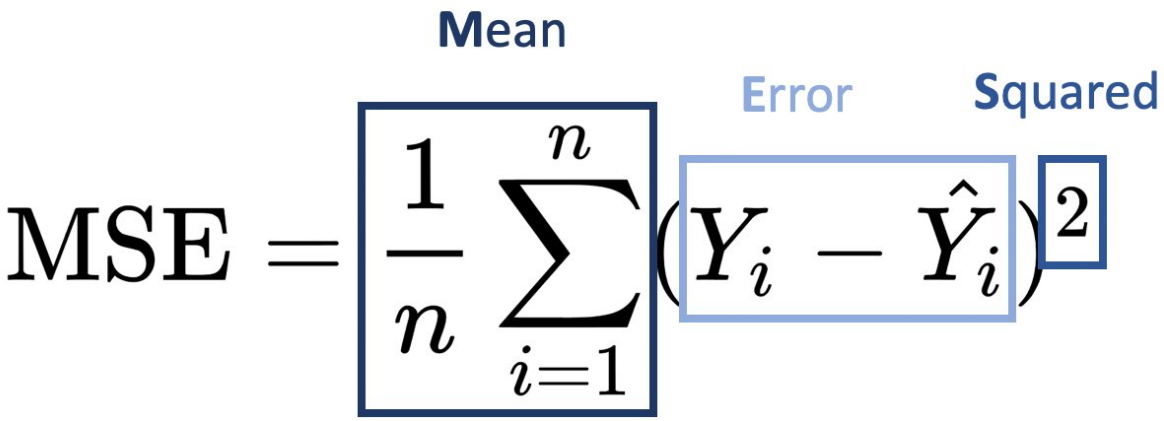
Mean Squared Error (MSE) is a regression metric that measures the average of the squared differences between the predicted and actual values. It is a **sensitive** metric, meaning that it is more sensitive to outliers than other metrics, such as MAE.

- its also used as **Loss Function**

Formula:

```
MSE = 1/n * Σ(y_true - y_pred)^2
```

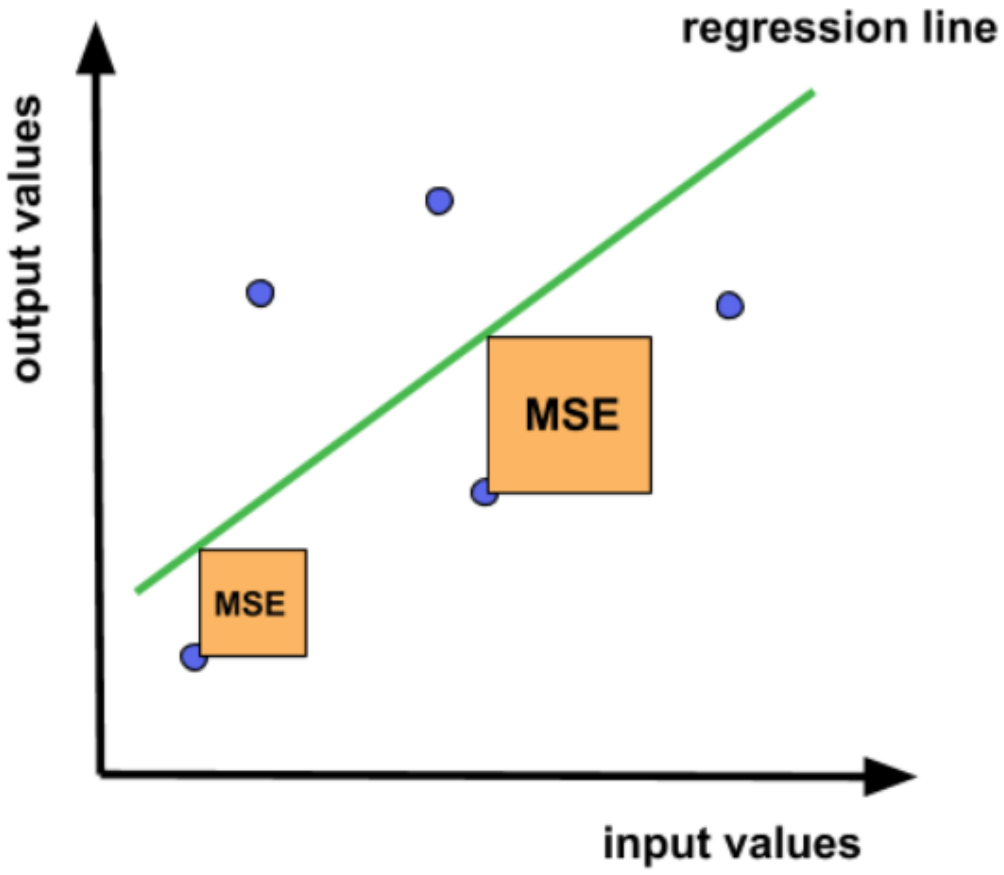
- **y_true:** The actual value of the target variable.
- **y_pred:** The predicted value of the target variable.
- **n:** The number of observations.



Interpretation:

The MSE is the average squared distance between the predicted and actual values. A lower MSE indicates that the model is closer to the actual values, while a higher MSE indicates that the model is further away from the actual values.

Image:



Example:

Let's say we have a regression model that predicts house prices. The actual house prices are in the range of $100,000 to 500,000$. If the MSE for the model is $100,000$, then this means that the model is, on average, 100,000 away from the actual house prices, squared.

How to choose MSE:

The MSE metric is a good choice when the goal is to minimize the overall error in the model, even if it means that some outliers are penalized more heavily. For example, if the target variable is a continuous variable with a narrow range of values, then the MSE metric may be a good choice.

Other metrics:

There are other regression metrics that can be used to evaluate the performance of a model, such as MAE and RMSE. The choice of metric depends on the specific problem and the desired outcome

Advantages and disadvantages of MSE:

Advantages:

- **Sensitive to large errors:** MSE is more sensitive to large errors than other metrics, such as MAE. This means that it is better at detecting models that are consistently over- or under-estimating the target variable.
- **Differentiable:** MSE is differentiable, which means that it can be used with gradient descent optimization algorithms.

Disadvantages:

- **Not robust to outliers:** MSE is not as robust to outliers as other metrics, such as MAE. This means that it may be affected by a few extreme values in the data.
- **Not scale-independent:** MSE is not scale-independent, which means that it is affected by the units of the data. This makes it a less good choice for comparing models that are trained on different datasets with different scales.

Overall, MSE is a good choice for evaluating the performance of a regression model when the goal is to minimize the overall error in the model, even if it means that some outliers are penalized more heavily. However, it is important to be aware of the limitations of MSE, such as its lack of robustness to outliers.

Here is a table summarizing the advantages and disadvantages of MSE:

Advantage	Disadvantage
Sensitive to large errors	Not robust to outliers
Differentiable	Not scale-independent

RMSE (Root Mean Squared Error)

Root Mean Squared Error (RMSE) is a regression metric that measures the **average magnitude** of the errors between the predicted and actual values. It is a **sensitive** metric, meaning that it is affected by outliers.

Formula:

```
RMSE = √(1/n * Σ(y_true - y_pred)^2)
```

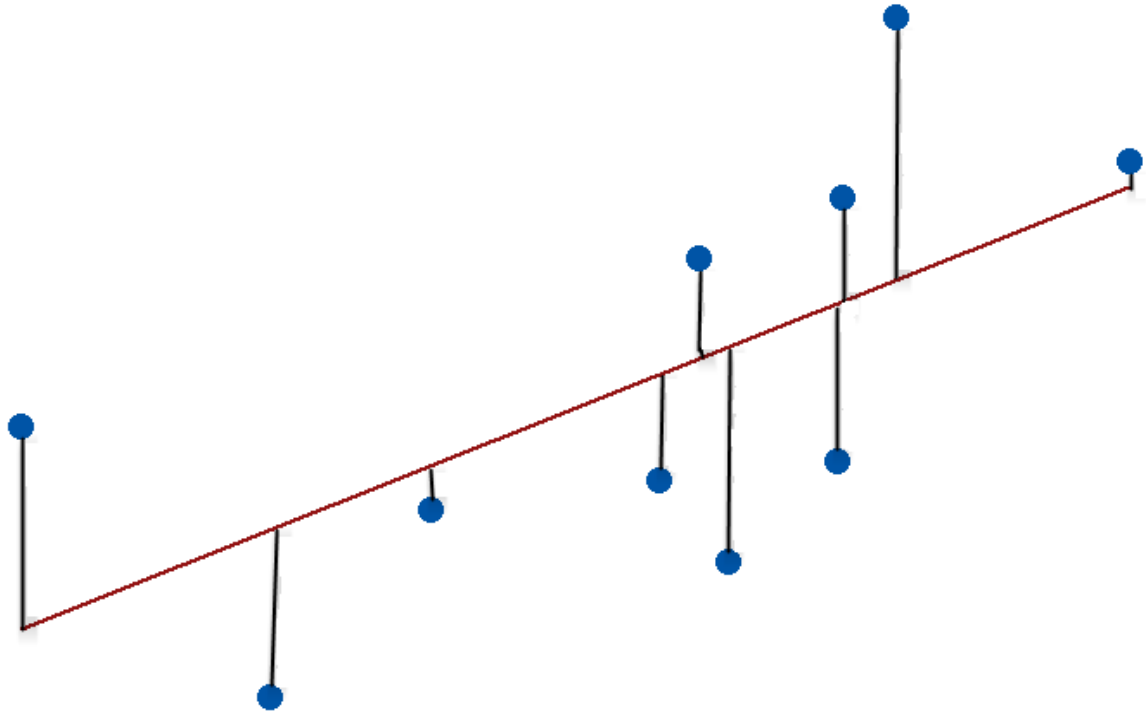
- **y_true:** The actual value of the target variable.
- **y_pred:** The predicted value of the target variable.
- **n:** The number of observations.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

Interpretation:

The RMSE is the square root of the average squared error. A lower RMSE indicates that the model is closer to the actual values, while a higher RMSE indicates that the model is further away from the actual values.

Image:



Example:

Let's say we have a regression model that predicts house prices. The actual house prices are in the range of $100,000 to 500,000$. If the RMSE for the model is $50,000, then this means that the model is, on average, 50,000$ away from the actual house prices, but the errors are not as sensitive to outliers as they would be with MAE.

How to choose RMSE:

The RMSE metric is a good choice when the goal is to minimize the **absolute fit** of the model. For example, if the target variable is a continuous variable with a wide range of values, then the RMSE metric may be a good choice.

Other metrics:

There are other regression metrics that can be used to evaluate the performance of a model, such as MAE and R-squared. The choice of metric depends on the specific problem and the desired outcome.

Advantages of RMSE:

- **Sensitive to large errors:** RMSE is more sensitive to large errors than other metrics, such as MAE. This means that it is better at detecting models that are consistently over- or under-estimating the target variable.
- **Scale-independent:** RMSE is scale-independent, which means that it is not affected by the units of the data. This makes it a good choice for comparing models that are trained on different datasets with different scales.

Disadvantages of RMSE:

- **Not as robust to outliers:** RMSE is not as robust to outliers as other metrics, such as MAE. This means that it may be affected by a few extreme values in the data.
- **Not as easy to interpret:** RMSE is not as easy to understand and interpret as other metrics, such as MAE. It is the square root of the average squared error, which can be difficult to understand for non-technical audiences.

Overall, RMSE is a good choice for evaluating the performance of a regression model when the goal is to minimize the absolute fit of the model. However, it is important to be aware of the limitations of RMSE, such as its sensitivity to outliers and its difficulty to interpret.

Here is a table summarizing the advantages and disadvantages of RMSE:

Advantage	Disadvantage
Sensitive to large errors	Not as robust to outliers
Scale-independent	Not as easy to interpret

R2 (r_Squared)

R-squared (R2) is a regression metric that measures the proportion of variance in the target variable explained by the model. It is a **normalized** metric, meaning that it is not affected by the units of the data.

Formula:

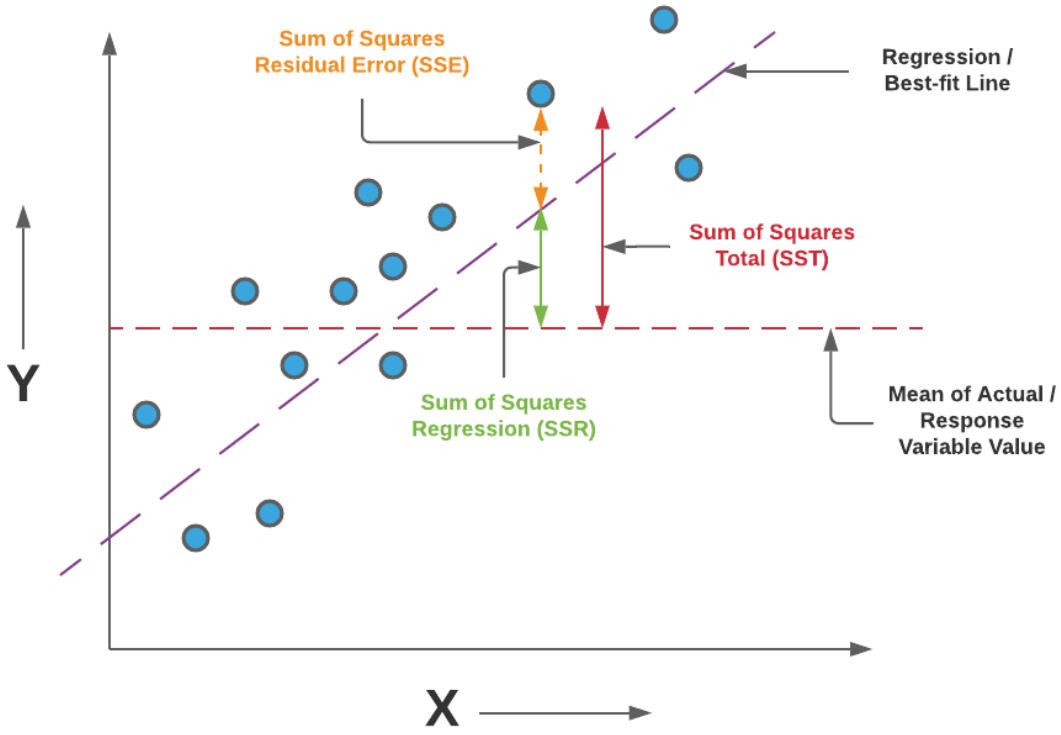
```
R^2 = 1 - (SSR/SST)
```

- **SSR:** The sum of squared residuals.
- **SST:** The total sum of squares.

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

Interpretation:

R2 is a percentage, and **it ranges from 0 to 1**. A higher R2 indicates that the model is better at explaining the variance in the target variable. A perfect model would have an R2 of 1, which means that the model perfectly explains the variance in the target variable.



Advantages:

- **Easy to understand and interpret:** R2 is easy to understand and interpret. It is simply the proportion of variance in the target variable that is explained by the model.
- **Normalized:** R2 is normalized, which means that it is not affected by the units of the data. This makes it a good choice for comparing models that are trained on different datasets with different scales.

Disadvantages:

- **Sensitive to outliers:** R2 is sensitive to outliers, meaning that a few extreme values in the data can significantly affect the value of R2.

- **Not a good measure of absolute fit:** R2 is not a good measure of absolute fit, meaning that it does not tell you how close the predicted values are to the actual values.

Overall, R2 is a good choice for evaluating the performance of a regression model when the goal is to measure the proportion of variance in the target variable that is explained by the model. However, it is important to be aware of the limitations of R2, such as its sensitivity to outliers.

Here is a table summarizing the advantages and disadvantages of R2:

Advantage	Disadvantage
Easy to understand and interpret	Sensitive to outliers
Normalized	Not a good measure of absolute fit

Adjusted R_Square

Adjusted R-squared is a metric that measures the proportion of variance in the target variable explained by the model, adjusted for the number of predictors used in the model. It is a modification of the R-squared metric that penalizes the model for adding additional predictors that do not significantly improve the fit of the model.

Formula:

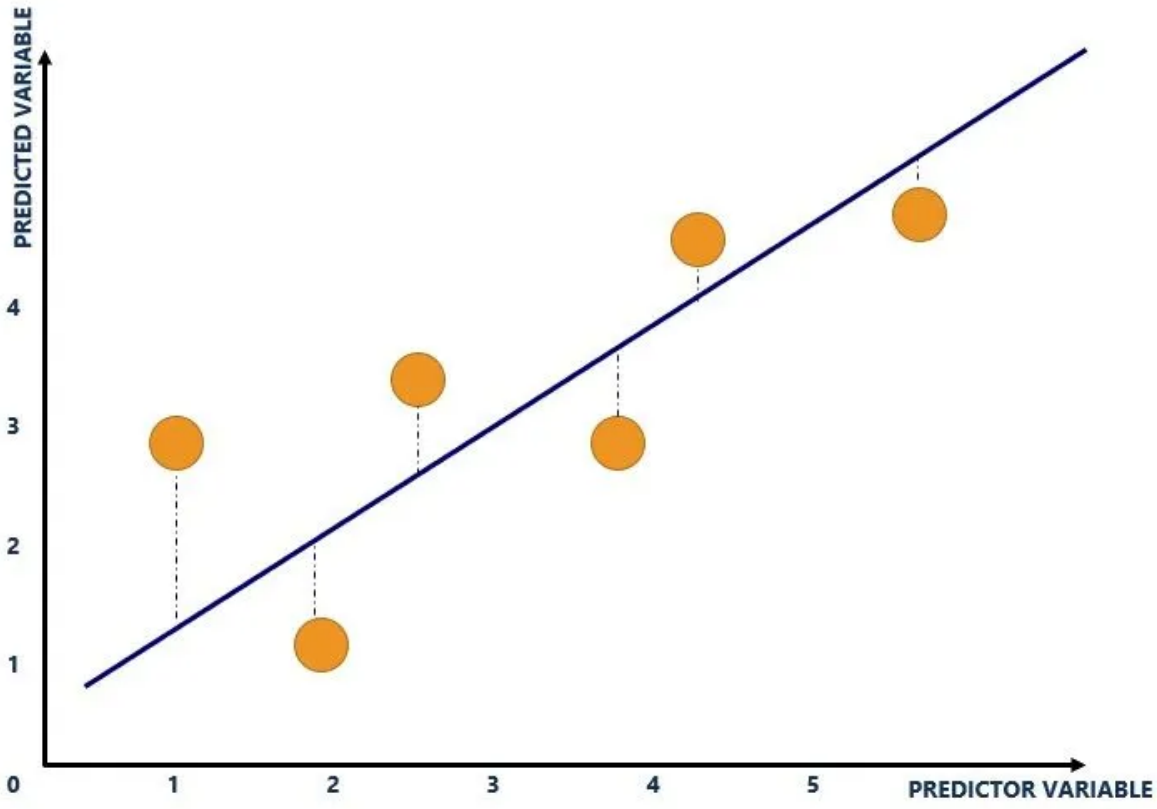
```
Adj. R2 = {1 - (1-R^2)(n-1) / (n-k-1)}
```

- **n:** the number of data points in our dataset
- **k:** the number of independent variables
- **R:** The R-squared values determined by the model.

$$Adjusted\ R^2 = \{1 - [\frac{(1-R^2)(n-1)}{(n-k-1)}]\}$$

Interpretation:

The Adj. R2 is a measure of how well the model fits the data, taking into account the number of predictors in the model. A higher Adj. R2 indicates that the model fits the data better.



Advantages:

- **Adjusts for the number of predictors:** Adj. R2 penalizes the model for adding additional predictors that do not significantly improve the fit of the model. This makes it a more reliable measure of the model's performance than R-squared, which can be inflated by adding irrelevant predictors.
- **Easy to interpret:** Adj. R2 is easy to interpret. It is simply a percentage that indicates the proportion of variance in the target variable that is explained by the model.

Disadvantages:

- **Sensitive to outliers:** Adj. R2 is sensitive to outliers, meaning that it can be affected by a few extreme values in the data.
- **Not scale-invariant:** Adj. R2 is not scale-invariant, meaning that it is affected by the units of the data. This makes it a less reliable measure of the model's performance when the data is measured in different units.

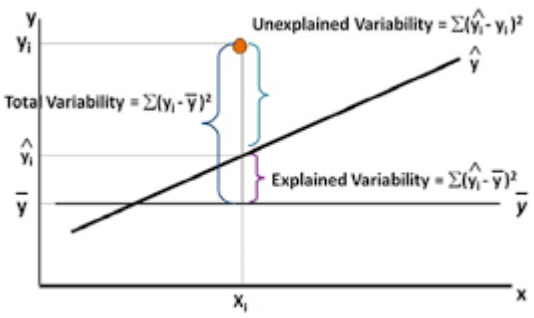
Overall, Adj. R2 is a good choice for evaluating the performance of a regression model when the goal is to measure the proportion of variance in the target variable that is explained by the model, taking into account the number of predictors in the model. However, it is important to be aware of the limitations of Adj. R2, such as its sensitivity to outliers and its lack of scale invariance.

Here is a table summarizing the advantages and disadvantages of Adj. R2:

Advantage	Disadvantage
Adjusts for the number of predictors	Sensitive to outliers
Easy to interpret	Not scale-invariant

Difference Between R-squared. and Adjusted R-squared.

- **R-squared** is a metric that measures the proportion of variance in the target variable explained by the model. It is a measure of how well the model fits the data. R-squared ranges from 0 to 1, with 1 indicating a perfect fit and 0 indicating no fit.
- **Adjusted R-squared** is a metric that measures the proportion of variance in the target variable explained by the model, adjusted for the number of predictors used in the model. It is a modification of the R-squared metric that penalizes the model for adding additional predictors that do not significantly improve the fit of the model.



The main difference between R-squared and Adjusted R-squared is that Adjusted R-squared takes into account the number of predictors in the model. This means that Adjusted R-squared is a more reliable measure of the model's performance than R-squared, which can be inflated by adding irrelevant predictors.

- **For example**, if you have a model with 2 predictors and an R-squared of 0.8, this means that the model explains 80% of the variance in the target variable. However, if you add a third predictor that does not significantly improve the fit of the model, the R-squared may increase to 0.85. However, the Adjusted R-squared will not increase as much, because it takes into account the fact that you have added an additional predictor.

In general, Adjusted R-squared is a better measure of the model's performance than R-squared, especially when the model has a large number of predictors. However, R-squared is still a useful metric, and it can be used to compare models with different numbers of predictors.

Here is a table summarizing the difference between R-squared and Adjusted R-squared:

Metric	Description
R-squared	Measures the proportion of variance in the target variable explained by the model.

Metric	Description
Adjusted R-squared	Measures the proportion of variance in the target variable explained by the model, adjusted for the number of predictors used in the model.
Penalizes the model for adding additional predictors that do not significantly improve the fit of the model.	No
More reliable measure of the model's performance	Not as reliable as Adjusted R-squared, but still a useful

- The key difference between R-squared and Adjusted R-squared is that Adjusted R-squared accounts for the complexity of the model by considering the number of predictors. It provides a more accurate measure of the model's goodness of fit and helps in comparing models with different numbers of predictors.

In summary, R-squared measures the proportion of variance explained by the independent variables without considering the model's complexity, while Adjusted R-squared adjusts for the number of predictors, providing a more reliable measure of the model's explanatory power.

In [7]:
```
## Code

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df =pd.read_csv("D:\\datascience\\Nitish sir\\Machine Learning\\placement.csv"

df
```

Out[7]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25
3	7.42	3.67
4	6.94	3.57
...	...	...
195	6.93	2.46
196	5.89	2.57
197	7.21	3.24
198	7.63	3.96
199	6.22	2.33

200 rows × 2 columns

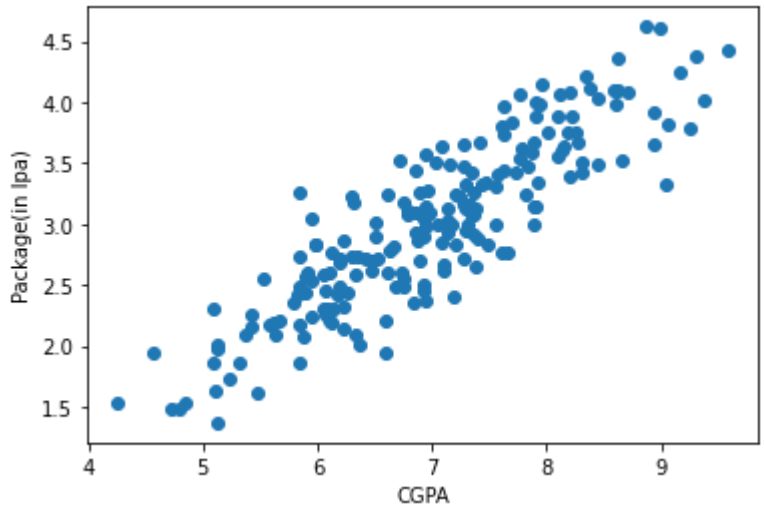
In [8]:
```
df.shape
```

Out[8]:
```
(200, 2)
```

In [9]:
```
# Plot a scatter plot of CGPA vs Package using the provided DataFrame.
plt.scatter(df['cgpa'],df['package'])

plt.xlabel('CGPA')
plt.ylabel('Package(in lpa)')
```

Out[9]:
```
Text(0, 0.5, 'Package(in lpa)')
```



In [10]:
```
# Extracting the first column of the DataFrame into X
X = df.iloc[:, 0:1]

# Extracting the last column of the DataFrame into y
y = df.iloc[:, -1]
```

In [12]:
```
X
```

Out[12]:

	cgpa
0	6.89
1	5.12
2	7.82
3	7.42
4	6.94
...	...
195	6.93
196	5.89
197	7.21
198	7.63
199	6.22

200 rows × 1 columns

In [13]:
```
y
```

Out[13]:
```
0      3.26
1      1.98
2      3.25
3      3.67
4      3.57
       ...
195    2.46
196    2.57
197    3.24
198    3.96
199    2.33
Name: package, Length: 200, dtype: float64
```

In [14]:
```
from sklearn.model_selection import train_test_split

# Splitting the data into training and testing sets
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_stat
```

In [15]:
```
from sklearn.linear_model import LinearRegression

# Create an instance of LinearRegression
lr = LinearRegression()

# Fit the model using the training data
lr.fit(X_train, y_train)
```

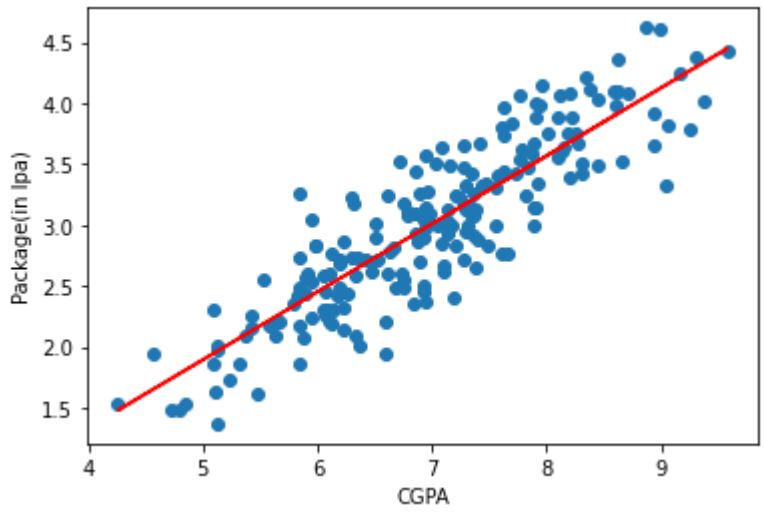
Out[15]:
```
LinearRegression
LinearRegression()
```

In [16]:
```
# Plot a scatter plot of 'cgpa' versus 'package' and overlay the linear regres
plt.scatter(df['cgpa'],df['package'])

#line
plt.plot(X_train,lr.predict(X_train),color='red')

#labels
plt.xlabel('CGPA')
plt.ylabel('Package(in lpa)')
```

Out[16]:
```
Text(0, 0.5, 'Package(in lpa)')
```



In [17]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [18]:
```
# Predict using the linear regression model on the test data
lr.predict(X_test)
```

Out[18]:
```
array([3.89111601, 3.09324469, 2.38464568, 2.57434935, 1.6537286 ,
       1.77647803, 2.07219258, 2.93143862, 3.76278706, 2.93701814,
       4.09197872, 3.51170867, 2.97049525, 2.40138424, 3.18809652,
       3.46707251, 1.94386362, 3.24389172, 2.97607477, 3.41685683,
       2.55761079, 3.16577844, 2.85890486, 3.12114229, 3.68467378,
       2.8700639 , 3.49497011, 3.34432308, 3.91901361, 1.96060218,
       3.65119666, 3.2104146 , 3.74046898, 2.7863711 , 2.78079158,
       3.27178932, 3.52844723, 2.61340599, 2.65804215, 2.71383735])
```

In [19]:
```
y_pred =lr.predict(X_test)
```

In [20]:
```
#Actual Values
y_test.values
```

Out[20]:
```
array([4.1 , 3.49, 2.08, 2.33, 1.94, 1.48, 1.86, 3.09, 4.21, 2.87, 3.65,
       4.  , 2.89, 2.6 , 2.99, 3.25, 1.86, 3.67, 2.37, 3.42, 2.48, 3.65,
       2.6 , 2.83, 4.08, 2.56, 3.58, 3.81, 4.09, 2.01, 3.63, 2.92, 3.51,
       1.94, 2.21, 3.34, 3.34, 3.23, 2.01, 2.61])
```

In [21]:
```
# MAE

print("MAE",mean_absolute_error(y_test,y_pred))


# Result - lpa Units
```

```
MAE 0.2884710931878175
```

Explanation:

It calculates the Mean Absolute Error (MAE) between the **predicted values (y_pred)** and the **actual values (y_test)** in a regression problem. The MAE value is then printed.

- **mean_absolute_error** is a function or method that calculates the MAE between two sets of values.
- **y_test** represents the actual or true values of the target variable in the regression problem.
- **y_pred** represents the predicted values of the target variable obtained from the regression model.

The "mean_absolute_error" function computes the absolute difference between each predicted value and its corresponding actual value. It then calculates the average of these absolute differences to obtain the MAE.

The output of the code would be:

MAE 0.2884710931878175

- "MAE" is a label indicating the metric being printed.

The value 0.2884710931878175 is the computed MAE between the predicted and actual values. It represents the average absolute difference between the predicted and actual values in the regression problem. **The lower the MAE value, the better the model's performance, as it indicates smaller prediction errors on average.**

Overall, this code snippet allows you to calculate and print the MAE value as a measure of the model's performance in terms of the absolute difference between predicted and actual values.

In [24]:
```
# MSE

print("MSE",mean_squared_error(y_test,y_pred))


# Result - Not in lpa Units
```

```
MSE 0.12129235313495527
```

Explanation :

It calculates the Mean Squared Error (MSE) between the predicted values (y_pred) and the actual values (y_test) in a regression problem. The MSE value is then printed.

- **mean_squared_error** is a function or method that calculates the MSE between two sets of values.
- **y_test** represents the actual or true values of the target variable in the regression problem.
- **y_pred** represents the predicted values of the target variable obtained from the regression model.

The mean_squared_error function computes the squared difference between each predicted value and its corresponding actual value. It then calculates the average of these squared differences to obtain the MSE.

The output of the code would be:

```
MSE 0.12129235313495527
```

- "MSE" is a label indicating the metric being printed.
- The value 0.12129235313495527 is the computed MSE between the predicted and actual values. It represents the average squared difference between the predicted and actual values in the regression problem. **The lower the MSE value, the better the model's performance, as it indicates smaller prediction errors on average.**

Overall, this code snippet allows you to calculate and print the MSE value as a measure of the model's performance in terms of the squared difference between predicted and actual values.

In [25]:
```
# RMSE

print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))


# Results - in Lpa Units
```

```
RMSE 0.34827051717731616
```

Explanation :

It calculates the Root Mean Squared Error (RMSE) between the predicted values (y_pred) and the actual values (y_test) in a regression problem. The RMSE value is then printed.

- **mean_squared_error** is a function or method that calculates the MSE between two sets of values.
- **y_test** represents the actual or true values of the target variable in the regression problem.
- **y_pred** represents the predicted values of the target variable obtained from the regression model.
- **np.sqrt** is a function from the NumPy library that calculates the square root of a given value.

The mean_squared_error function computes the squared difference between each predicted value and its corresponding actual value. It then calculates the average of these squared differences to obtain the MSE. Finally, `np.sqrt` is used to take the square root of the MSE, resulting in the RMSE.

The output of the code would be:

```
RMSE 0.34827051717731616
```

- "RMSE" is a label indicating the metric being printed.
- The value 0.34827051717731616 is the computed RMSE between the predicted and actual values. It represents the square root of the average squared difference between the predicted and actual values in the regression problem. **The lower the RMSE value, the better the model's performance, as it indicates smaller prediction errors on average.**

Overall, this code snippet allows you to calculate and print the RMSE value as a measure of the model's performance in terms of the root of the squared difference between predicted and actual values.

In [27]:
```
# r2_Score

print("r2_Score:",r2_score(y_test,y_pred))
```

```
r2_Score: 0.780730147510384
```

Explanation:

it calculates the R-squared (R^2) between the predicted values (y_pred) and the actual values (y_test) in a regression problem. The R^2 value is then printed.

- **r2_score** is a function or method that calculates the R-squared between two sets of values.
- **y_test** represents the actual or true values of the target variable in the regression problem.
- **y_pred** represents the predicted values of the target variable obtained from the regression model.

The r2_score function computes the R-squared, which measures the proportion of the total variance in the dependent variable that can be explained by the independent variables in the regression model.

```
r2: 0.780730147510384
```

- "r2:" is a label indicating the metric being printed.
- The value 0.780730147510384 is the computed R^2 between the predicted and actual values. It represents the proportion of the variance in the dependent variable explained by the independent variables in the regression model. **R^2 ranges from 0 to 1, where 1 indicates that the model explains all the variability in the data.**

Overall, this code snippet allows you to calculate and print the R^2 value as a measure of the model's performance in explaining the variance in the dependent variable. A higher R^2 value indicates a better fit of the model to the data.

In [28]:
```
# r2

r2 = r2_score(y_test,y_pred)
```

In [29]:
```
# Adjusted r2

X_test.shape
```

Out[29]:
```
(40, 1)
```

In [30]:
```
# Formula

1 - ((1-r2)*(40-1)/(40-1-1))
```

Out[30]:
```
0.7749598882343415
```

Explanation:

the provided shape of the test data, it seems that you are calculating the Adjusted R-squared score manually using the formula:

Adjusted $R^2 = 1 - ((1 - R^2) * (n - 1) / (n - p - 1))$

where:

- R^2 is the R-squared score.
- n is the number of samples (40 in this case).
- p is the number of predictors or independent variables (1 in this case).

Let's break down the calculation using the provided values:

```
X_test.shape
(40, 1)
```

- The shape of `X_test` indicates that you have 40 samples and 1 predictor variable.

```
1 - ((1 - r2) * (40 - 1) / (40 - 1 - 1))
0.7749598882343415
```

- `r2` refers to the R-squared score, which is 0.780730147510384 as calculated earlier.
- Plugging in the values into the formula, we get 0.7749598882343415 as the calculated Adjusted R^2 score.

The Adjusted R-squared score is used to account for the number of predictors in the model, **providing a penalty for adding unnecessary variables that do not contribute significantly to the model's performance.** In this case, the calculated Adjusted R^2 score suggests that the model explains approximately 77.5% of the variance in the dependent variable, taking into account the single predictor variable and the number of samples.

Importance of Adjusted R-Squared

In [31]:
```
# Create a copy of the original DataFrame
new_df1 = df.copy()

# Add a new column 'random_feature' with random values between 0 and 1
new_df1['random_feature'] = np.random.random(200)

# Reorder the columns of the DataFrame
new_df1 = new_df1[['cgpa', 'random_feature', 'package']]

# Display the first few rows of the modified DataFrame
new_df1.head()
```

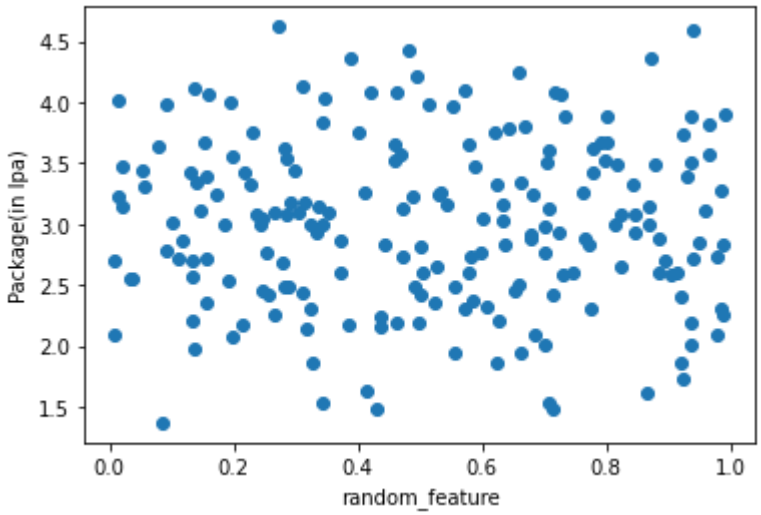
Out[31]:

	cgpa	random_feature	package
0	6.89	0.533412	3.26
1	5.12	0.137277	1.98
2	7.82	0.529477	3.25
3	7.42	0.789529	3.67
4	6.94	0.467272	3.57

In [32]:
```
# Random data is not linear

plt.scatter(new_df1['random_feature'],new_df1['package'])
plt.xlabel('random_feature')
plt.ylabel('Package(in lpa)')
```

Out[32]: Text(0, 0.5, 'Package(in lpa)')



In [45]:
```
 # Import necessary libraries

X = new_df1.iloc[:, 0:2]
y = new_df1.iloc[:, -1]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Create and train the linear regression model
random_df = LinearRegression()
random_df.fit(X_train, y_train)

# Make predictions on the test set
y_pred = random_df.predict(X_test)

# Calculate and print the R2 score
print("R2 score:", r2_score(y_test, y_pred))

# Calculate the R2 score for further use
random_r2 = r2_score(y_test, y_pred)
```

```
R2 score: 0.7670139044847327
```

In [40]:
```
# Here we can clearly see that R2 score has slightly  decreased from

#r2_Score: 0.780730147510384 (Old) - random_r2_score: 0.7670139044847327 (rand
```

In [47]:
```
# Adjusted R2_score

1 - ((1-random_r2)*(40-1)/(40-1-2))
```

Out[47]:
```
0.7544200614839075
```

In [ ]:
```
# After apply Random data , Adjusted r2 has slightly decreased from

# Adjusted r2_score : 0.77495988823434159 (Old) - 0.7544200614839075 (random_d
```

Adding new Column With Linear Relationship

In [41]:
```
new_df2 = df.copy()

new_df2['iq'] = new_df2['package'] + (np.random.randint(-12,12,200)/10)

new_df2 = new_df2[['cgpa','iq','package']]
```

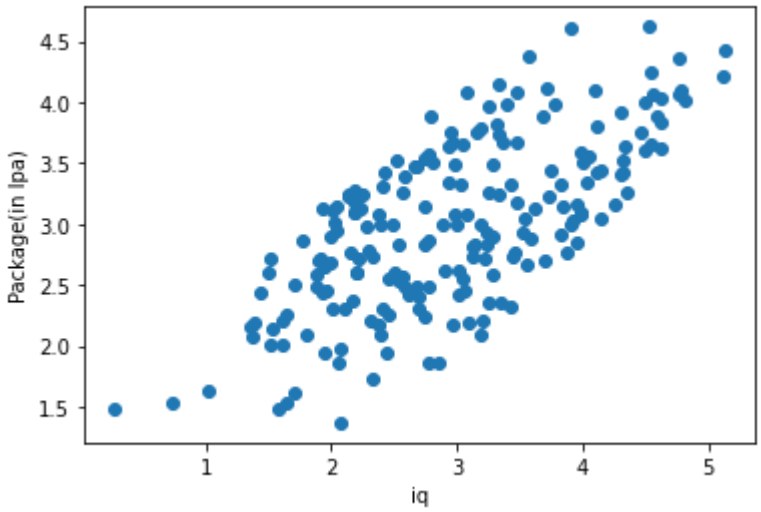
In [42]:
```
new_df2.head()
```

Out[42]:

	cgpa	iq	package
0	6.89	3.26	3.26
1	5.12	2.08	1.98
2	7.82	2.25	3.25
3	7.42	3.37	3.67
4	6.94	2.77	3.57

In [43]:
```
# Scatter plot of 'iq' vs 'package'
plt.scatter(new_df2['iq'],new_df2['package'])
plt.xlabel('iq')
plt.ylabel('Package(in lpa)')
```

Out[43]:
```
Text(0, 0.5, 'Package(in lpa)')
```



In [52]:
```
 # Import necessary libraries

X = new_df1.iloc[:, 0:2]
y = new_df1.iloc[:, -1]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Create and train the linear regression model
useful_df = LinearRegression()
useful_df.fit(X_train, y_train)

# Make predictions on the test set
y_pred = useful_df.predict(X_test)

# Calculate and print the R2 score
print("R2 score:", r2_score(y_test, y_pred))

# Calculate the R2 score for further use
useful_r2 = r2_score(y_test, y_pred)
```

```
R2 score: 0.7670139044847327
```

In [53]:
```
1 - ((1-useful_r2)*(40-1)/(40-1-2))
```

Out[53]:
```
0.7544200614839075
```

In [ ]:
```

```