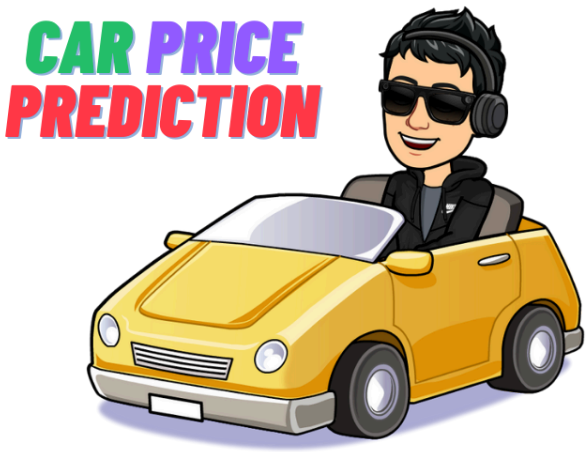
Content

        - R-square equal
        - Size of squares
        - Maxima and Minima of R-squared
- **Multivariate Linear Regression**
    - ASSESSMENT:
        - Life Expectancy
        - Coefficients & Intercept coding
- **Linear Regression - 3**
- **Model Interpretibility and Feature Importance**
    - ASSESSMENT:
        - Maxpower or year ?
- **ols implimenatation with stats model**
    - p value and feature importance
- **Assumptions of Linear Regression**
    - 1.Assumption of Linearity
    - 2.Features are not multi-collinear
    - VIF (Variance Inflation Factor)
    - 3.Errors are normally distributed
    - Error terms must be independent
    - Homoscedasticity: Error term have constant variance.
    - ASSESSMENT:
        - Performance with Multicollinearity
        - VIF
        - Curious case of VIF
        - Homoscedasticity in Linear Regression
        - VIF in Linear Regression
        - Multi-collinearity
        - Error Indepedence
        - Violations in Assumptions
        - Correlation of the Coefficients
        - Strong linear relationship
        - What is the slope ?
- **Impact of outliers on Linear Regression**
    - ASSESSMENT:
        - Interpreting Outliers
        - Outliers in Linear Regression

1. Problem Statement

- There is an automobile company **Cars24** from Japan which aspires to **enter** the **US market** by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.
- They want to **understand** the **factors affecting the pricing of cars** in the American market, since those may be very different from the Japanese market. Essentially, the company wants to know:
  - Which variables are significant in predicting the price of a car
  - How well those variables describe the price of a car
- Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the American market.



Business Objective:

- You as a **Data scientist** are required to apply some data science techniques to come up with **price of cars**. That should help the management to understand how exactly the prices vary with different features of car.

```
# importing numpy and pandas
import numpy as np
import pandas as pd
```

```
!gdown 1UpLnYA48Vy_lGUMMLG-uQE1gf_Je12Lh
```

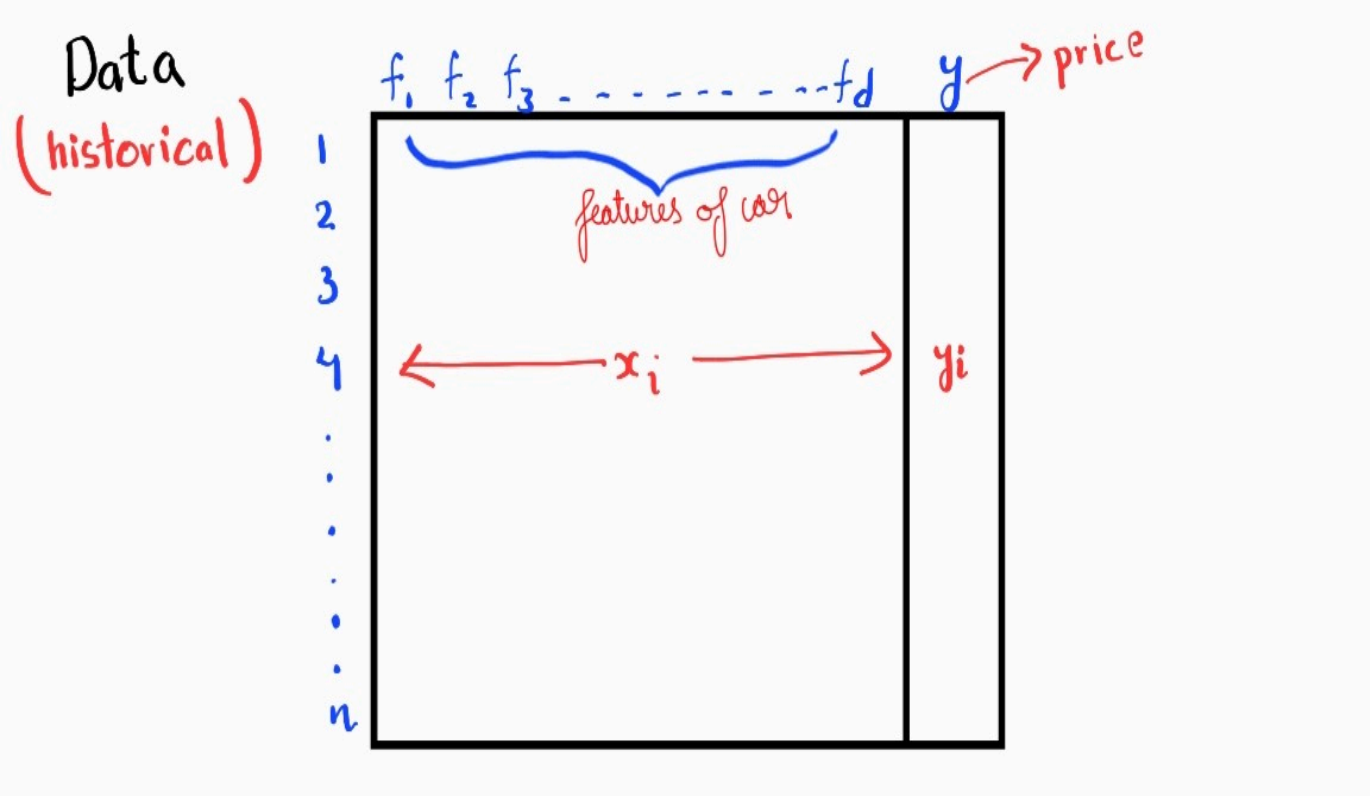
```
Downloading...
From: https://drive.google.com/uc?id=1UpLnYA48Vy_lGUMMLG-uQE1gf_Je12Lh
To: /content/cars24-car-price-clean.csv
100% 7.10M/7.10M [00:00<00:00, 42.4MB/s]
```

- This is the clean data of Cars24 that we will be using here, as you already gone through EDA & pre-processing steps.

```
df = pd.read_csv('cars24-car-price-clean.csv')
df.head()
```

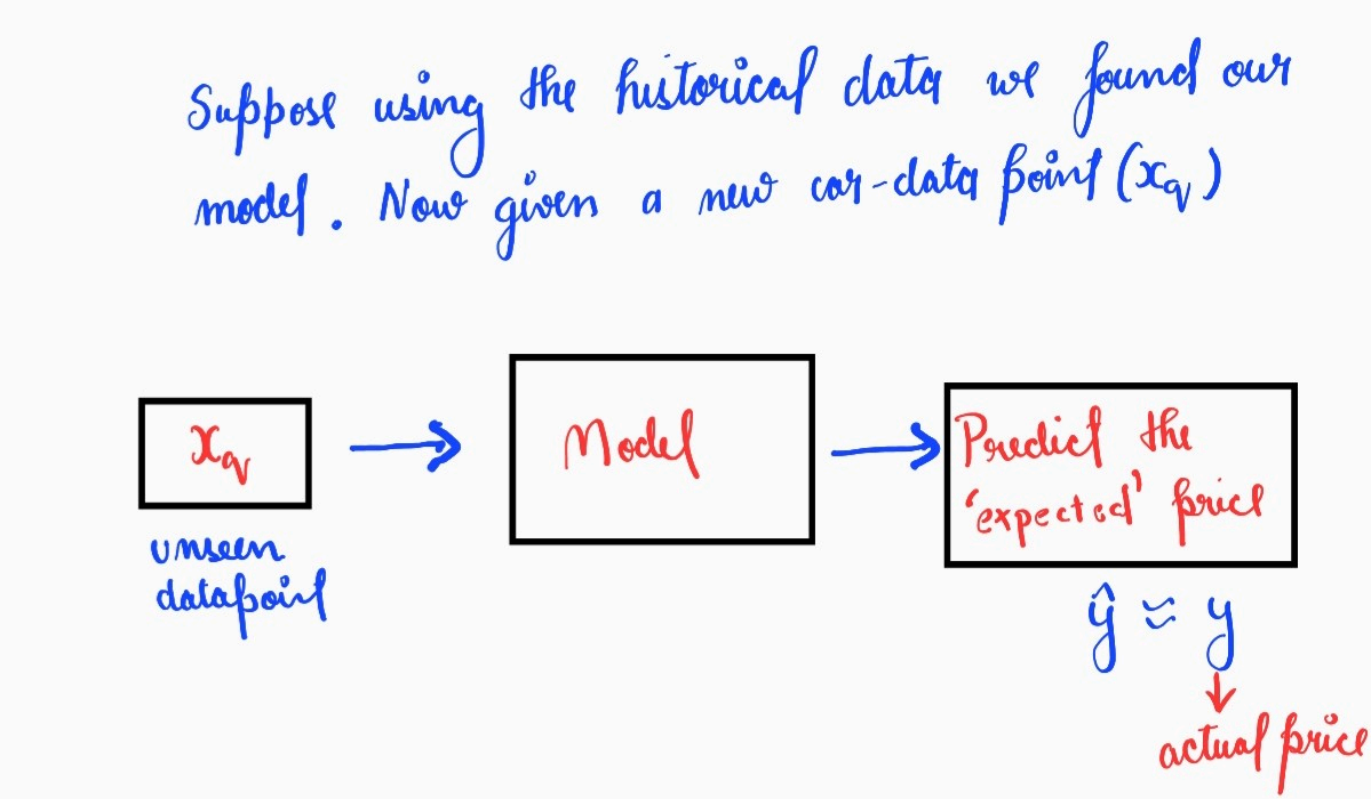
	selling_price	year	km_driven	mileage	engine	max_power	age	
0	-1.111046	-0.801317	1.195828	0.045745	-1.310754	-1.157780	0.801317	-0.43
1	-0.223944	0.450030	-0.737872	-0.140402	-0.537456	-0.360203	-0.450030	-0.32
2	-0.915058	-1.426990	0.035608	-0.582501	-0.537456	-0.404885	1.426990	-0.32
3	-0.892365	-0.801317	-0.409143	0.329620	-0.921213	-0.693085	0.801317	-0.43

2. Data Notation



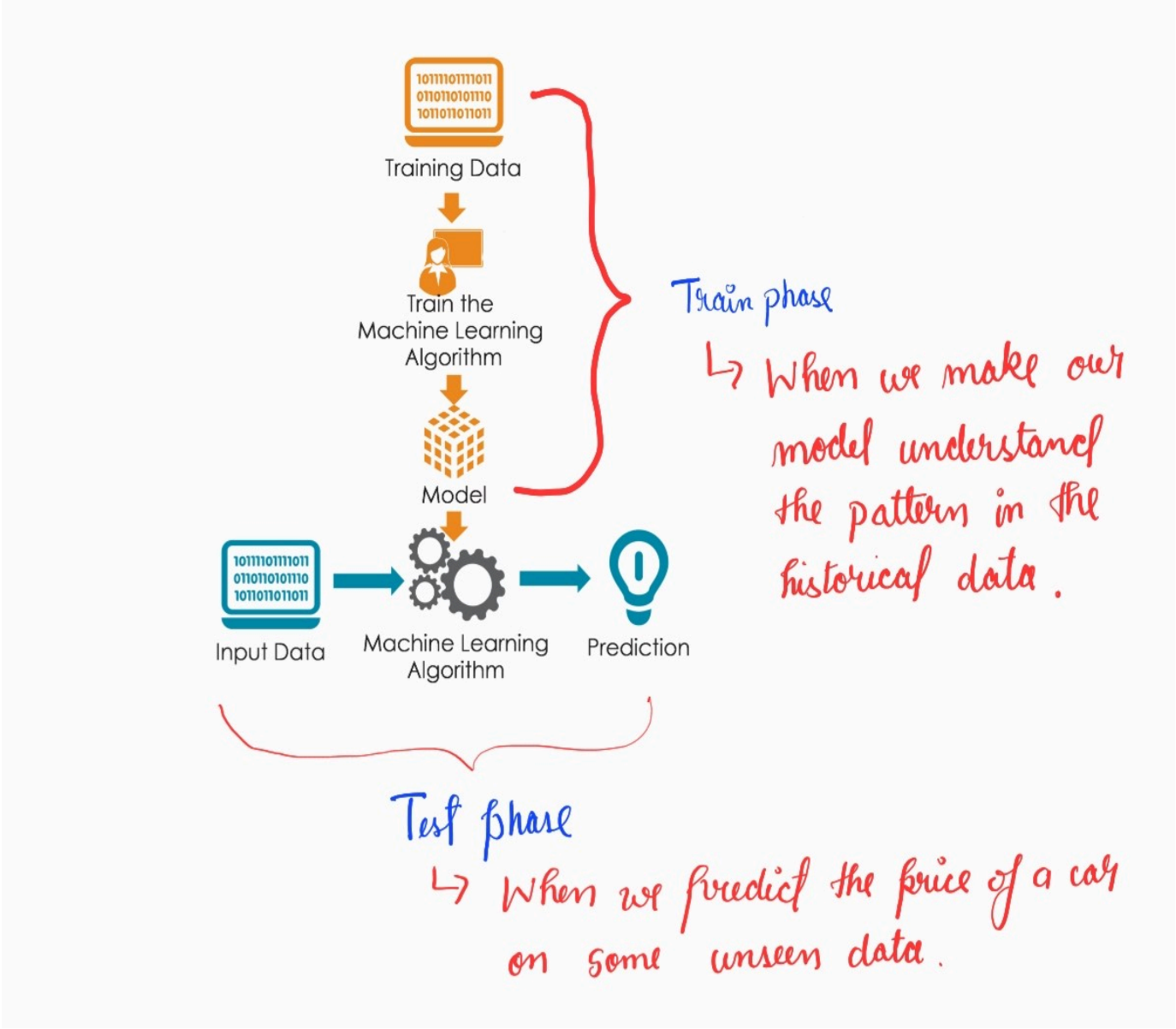
- Assume that we are given n cars datapoints. This is our historical data.
- We have bunch of features f1, f2 .... fd.
- xi represents n or d features corresponding to the ith car.
- yi is the price for the ith car.

- Using this historical data, we come up with a model .
- x here is d dimensional representation of the point.
- y is real number which tells the price of the car.
- Our goal in ML is to find this model.



- Assume that using the historical data, we found Model which maps our input x to our output value y .
- Now, given a new car, (xq), we pass it to model.
- Model should be able to prediction expected price.
- Predicted price is denoted using yq_hat. We use hat notation for predicted value.
- Our model is doing a very good job if yq_hat (predicted price) is very close to yq (actual price)

Let us understand the phases of a model



Question: Are we doing classification or regression?

Ans: Since we have to predict continuous numeric value, we are doing **regression task**

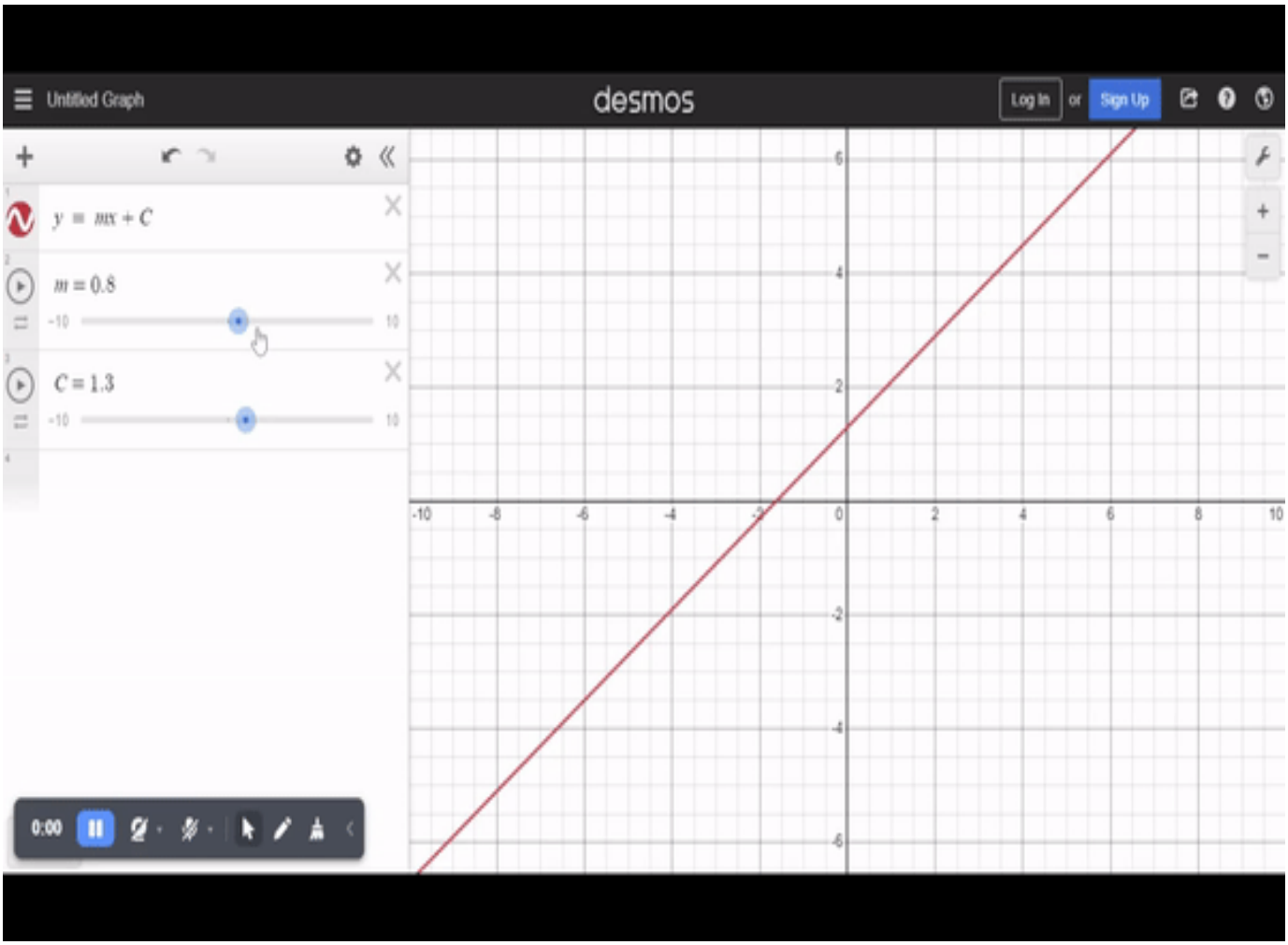
---

3. What approach should we take for this problem?

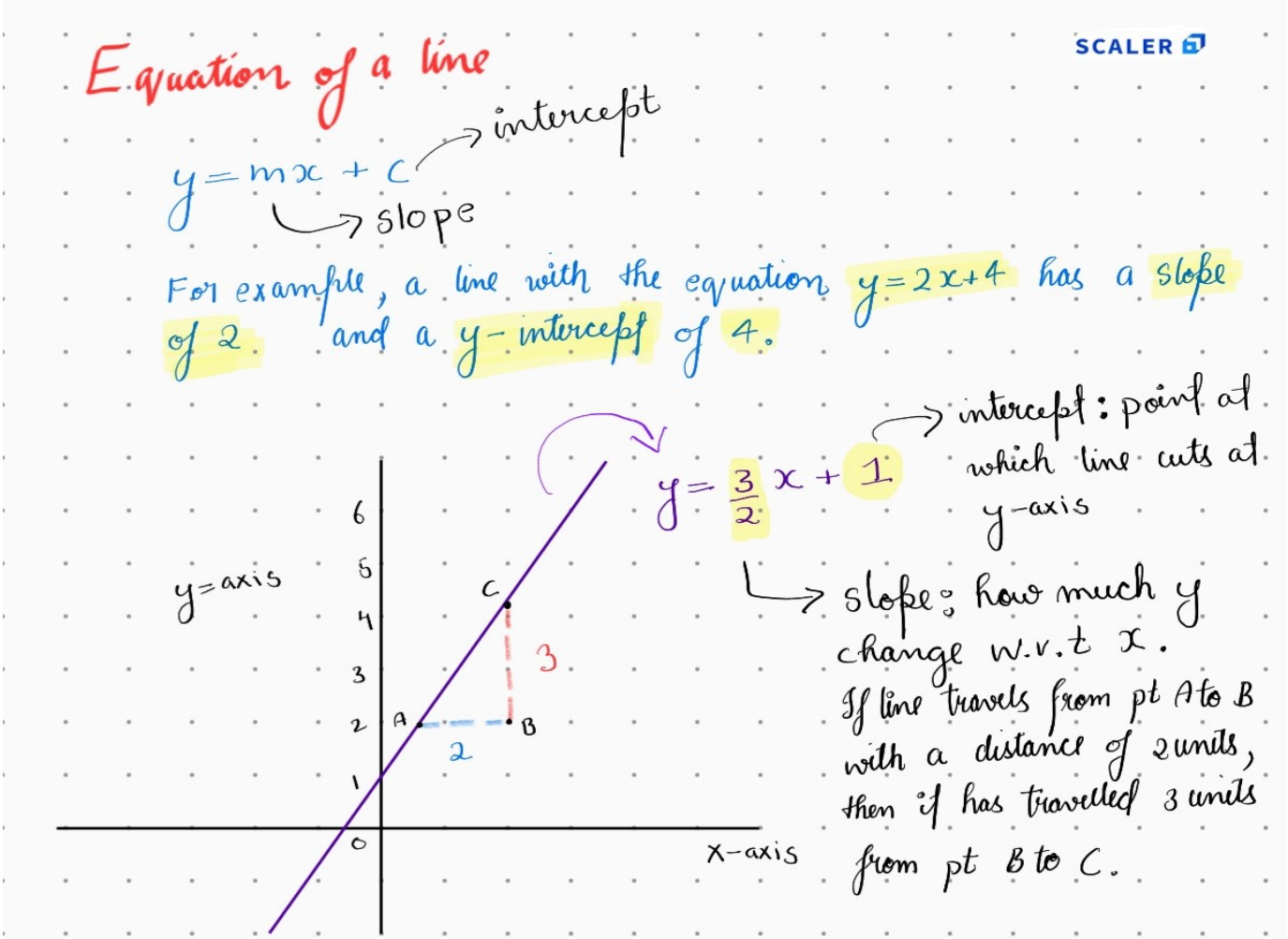
A simple approach is to check how **Y** depends on **X** and draw a trend line.

Before moving to a trend line, let's discuss what is a line ?

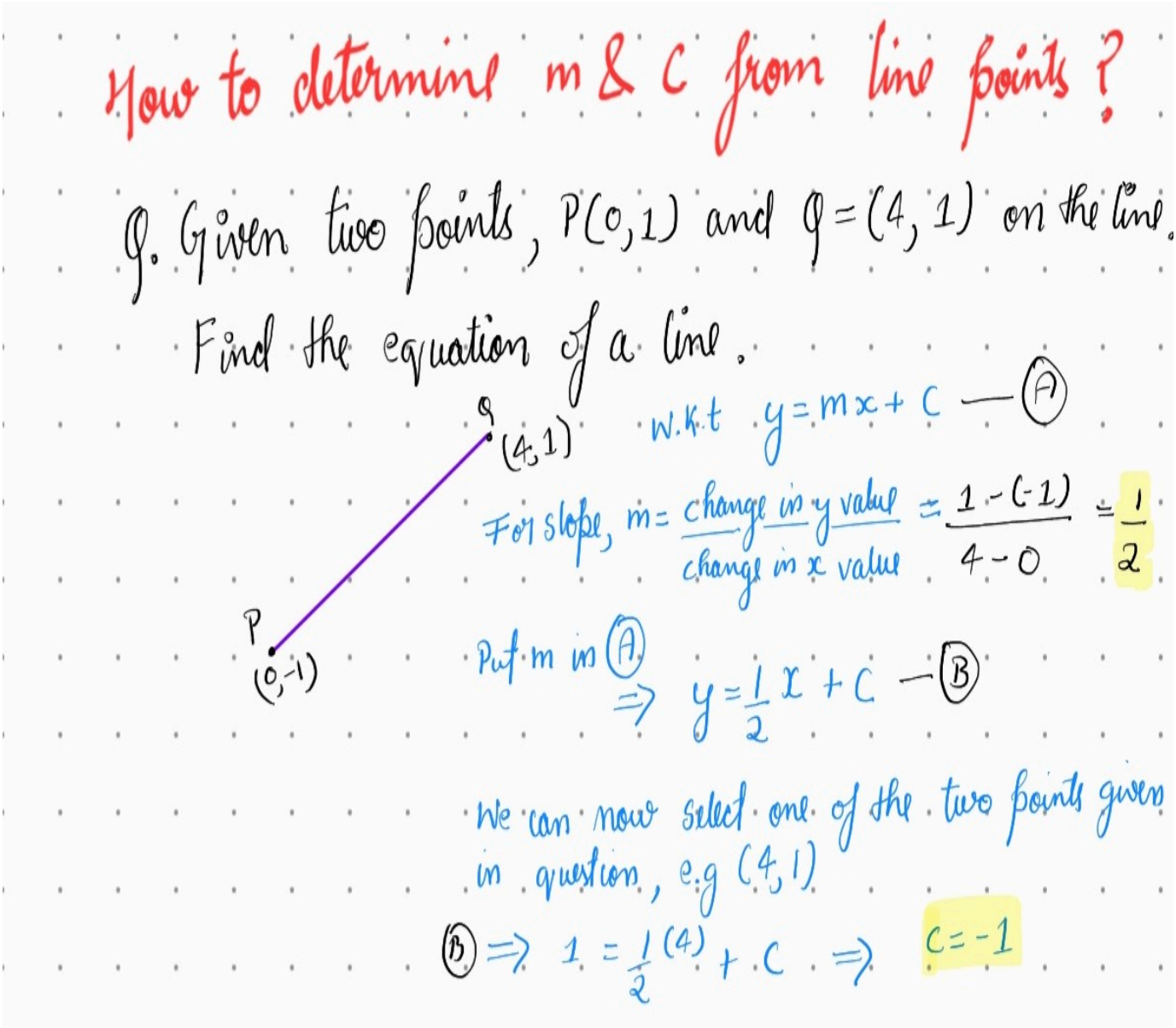
- Looking at the graph below , what is that we can find?



Equation of a line



- Note : If we observe the equation of a line we are predicting the value of y using the value of x. So y here is your dependent variable and x is independent variable.



Question : What will be the eqn of a line now that you have value m and c?

QUIZ-1

How many parameters do you need to estimate in a simple linear regression model (One independent variable)?

1. 1
2. 2
3. Can't Say

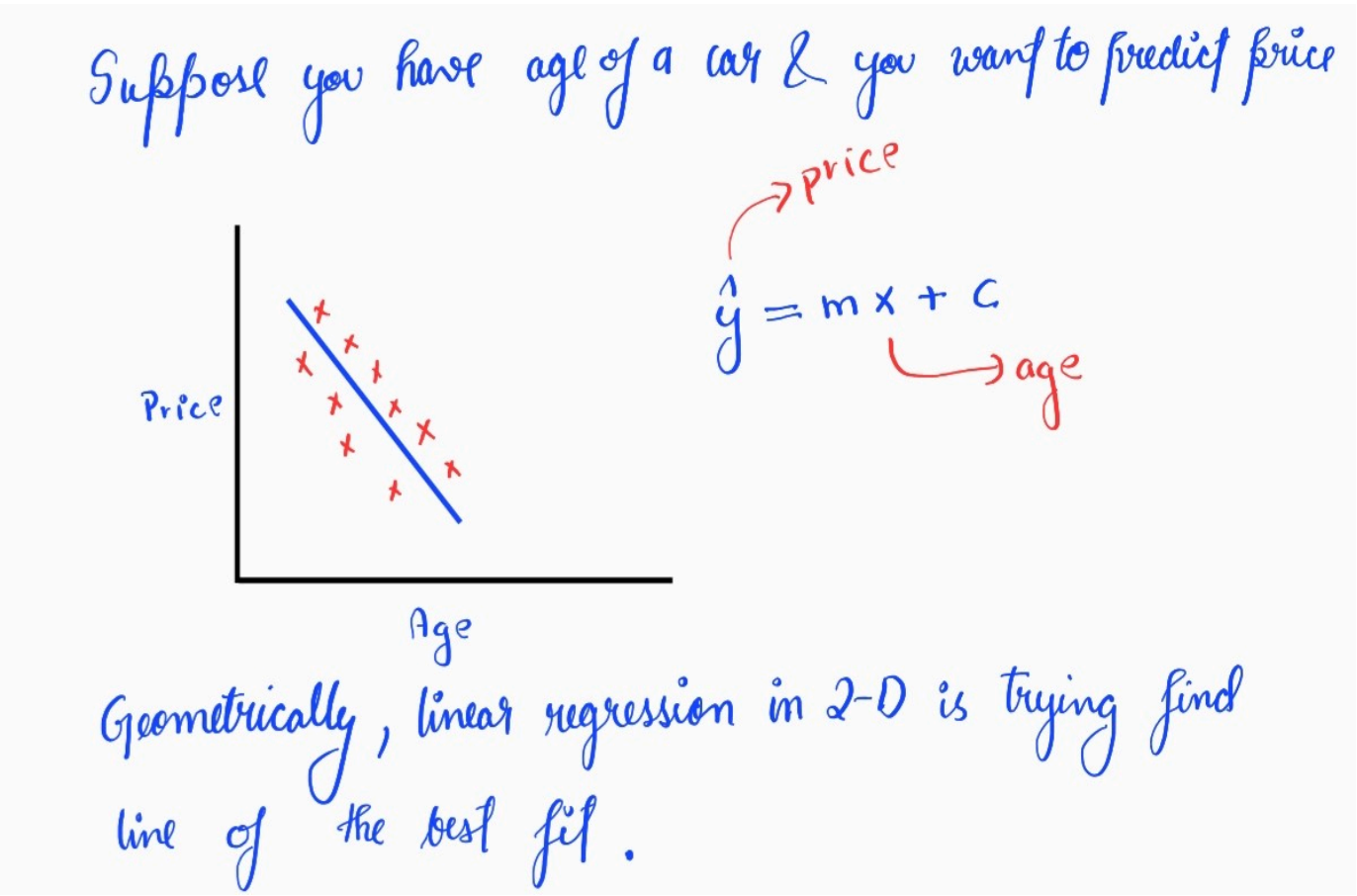
ANSWER

2. 2

EXPLANATION

- In simple linear regression, there is one independent variable so 2 coefficients (Y=a+bx).

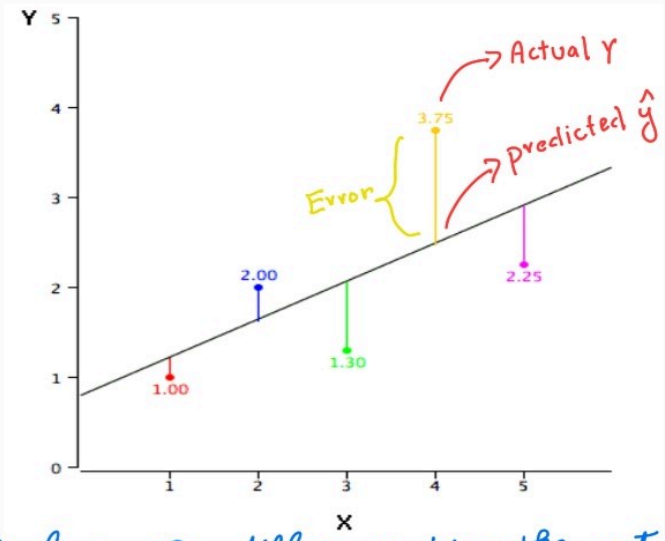
5. Geometric Intuition



Best fit line

- Most of the points are closer to 2nd line.
- 1st and 3rd lines are very far from the points.

Errors or Residual



- Error / Residual is the difference b/w the actual value y and the predicted value $\hat{y}$.
- We want to find a line which minimizes the diff b/w actual & predicted

$$e_i = y_i - \hat{y}_i$$

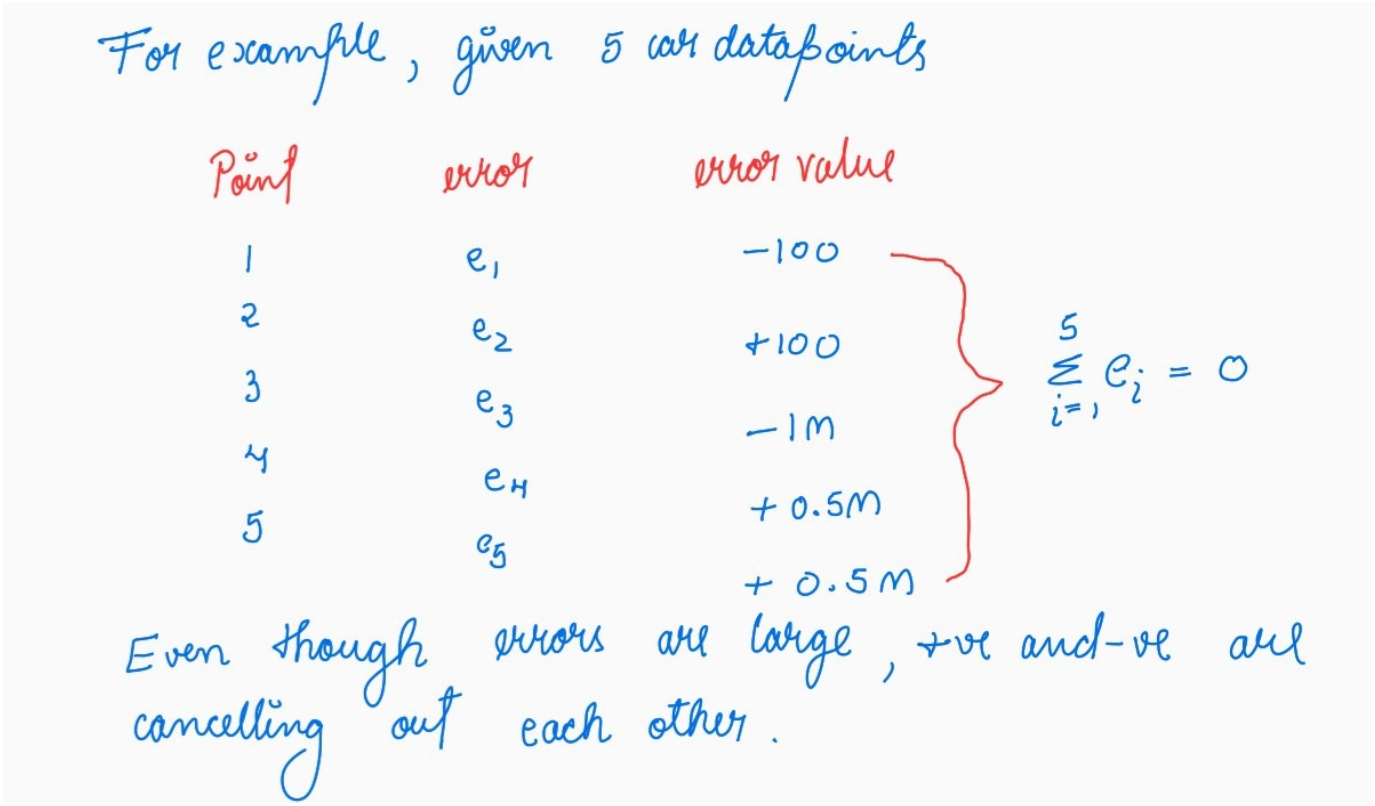
($\hat{y}_i$ → predicted; e_i → could be +ve/-ve; y_i → actual)

- We could take sum of all the errors

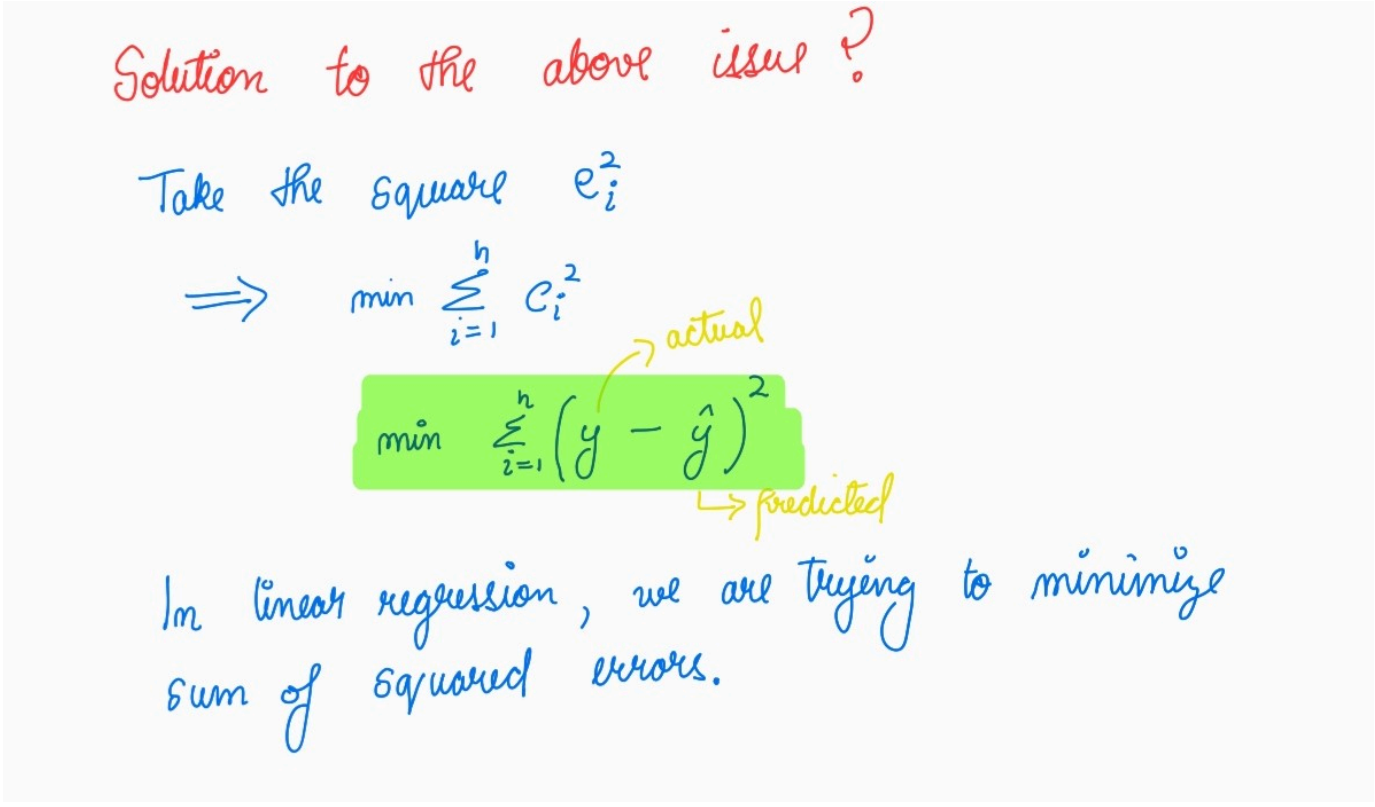
$$\min \sum_{i=1}^{n} e_i$$

- Error is the difference between the actual value of y and the predicted value of y.
- We could take sum of all the errors.

Question: But there is a issue while taking sum of errors? Any guess why?



- But there is an issue with this error. It can be both positive and negative.
- There can be case the error are large value but the sum of error is coming out to be 0.
- Positive and negative values are cancelling out
- Refer to the example in the image above.



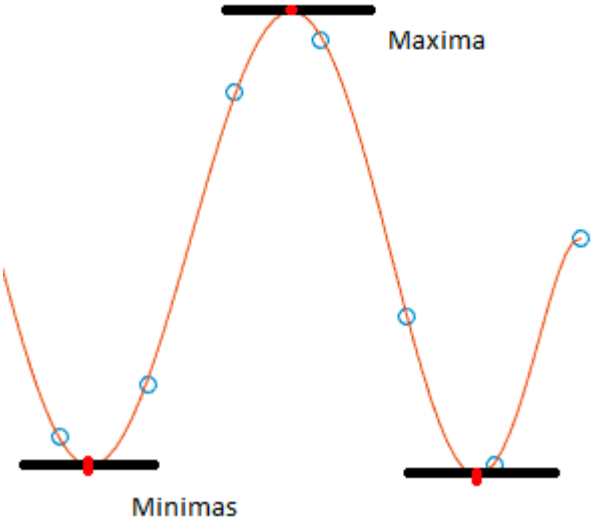
- We don't want our errors to cancel out.
- The solution to this problem is taking the squares of the errors.

- The method where we use **reducing the sum of the squares** is called **Ordinary Least Sqaures**.
- That means linear regression uses **Ordinary Least Sqaures**

> INSTRUCTOR NOTES
> why squared error

https://towardsdatascience.com/https-medium-com-chayankathuria-regression-why-mean-square-error-a8cad2a1c96f

- Absolute Errors:This loss function is not differentiable at 0.



Code implementation for Geometric intuition

- Here to show code implementaion of geometric intuition we will using same dataset but without clean part,so that we develop better understanding.
- We will predicting price of a car using Max_power

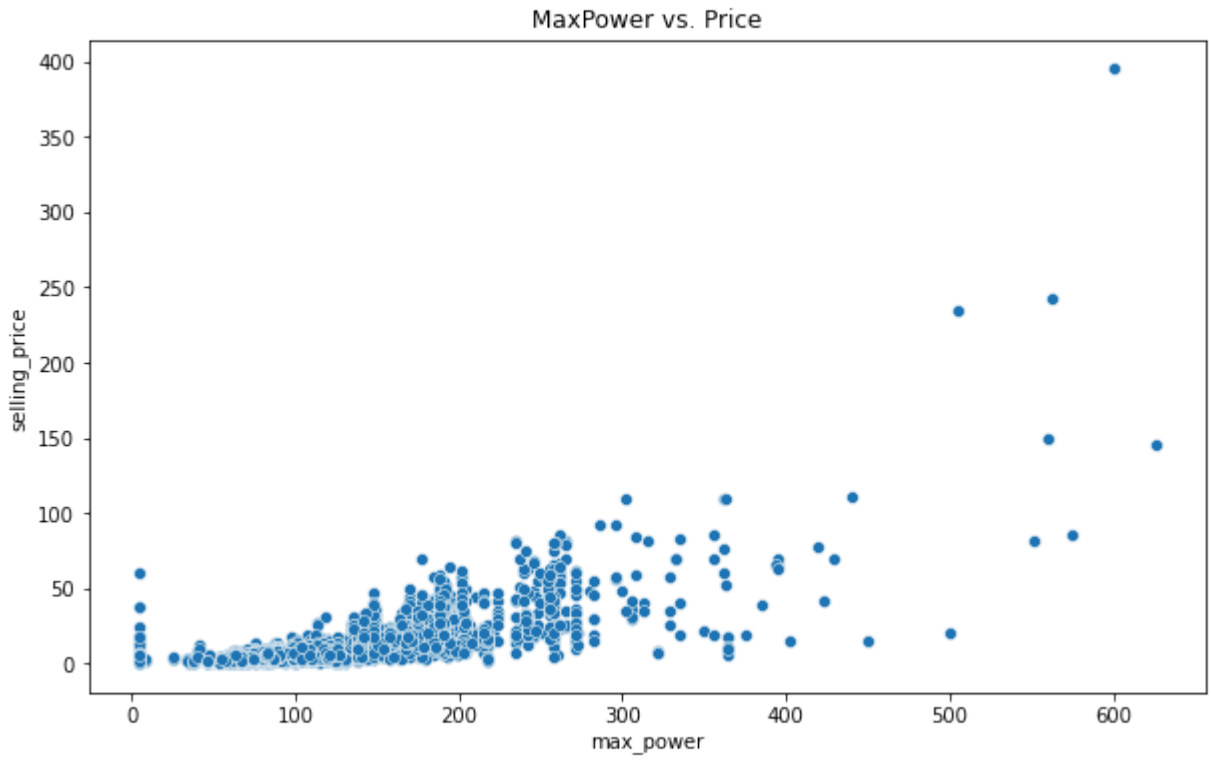
```
!gdown 1RRbLfbAPiihXP50KDQQc6Cr52-r0fsoF
```

```
Downloading...
From: https://drive.google.com/uc?id=1RRbLfbAPiihXP50KDQQc6Cr52-r0fsoF
To: /content/train-cars24-car-price.csv
100% 1.25M/1.25M [00:00<00:00, 26.9MB/s]
```

```
# Displaying our data
import pandas as pd
import numpy as np
data = pd.read_csv('/content/train-cars24-car-price.csv')
data.head()
```

	full_name	selling_price	year	seller_type	km_driven	fuel_type	transmission_
0	Maruti SX4 Zxi BSIII	2.85	2007.0	Individual	110000	Petrol	M
1	Hyundai i20 Sportz 1.4 CRDi	4.70	2012.0	Dealer	70000	Diesel	M
	Maruti Swift						

```
# using single feature to predict selling price of a car
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
plt.title('MaxPower vs. Price')
sns.scatterplot(data=data, x='max_power', y='selling_price');
```



Apart from a few exceptions, the points seem to form a line. We'll try and "fit" a line using this points, and use the line to predict price for a given max power. A line on the X&Y coordinates has

the following formula:

$y = wx + b$

The line is characterized two numbers: w (called "slope") and b (called "intercept").

Model

In the above case, the x axis shows "max_power" and the y axis shows "selling price of a car". Thus, we're assume the following relationship between the two:

$price = w \times maxpower + b$

We'll try determine w and b for the line that best fits the data.

- This technique is called *linear regression*, and we call the above equation a *linear regression model*, because it models the relationship between "max_power" and "price" as a straight line.
- The numbers w and b are called the *parameters* or *weights* of the model.
- The values in the "max_power" column of the dataset are called the *inputs* to the model and the values in the "selling_price" column are called "targets".

Let define a helper function `estimate_charges`, to compute $price$, given $maxpower$, w and b.

```
def estimate_charges(maxpower, w, b):
    return w * maxpower + b
```

The `estimate_charges` function is our very first *model*.

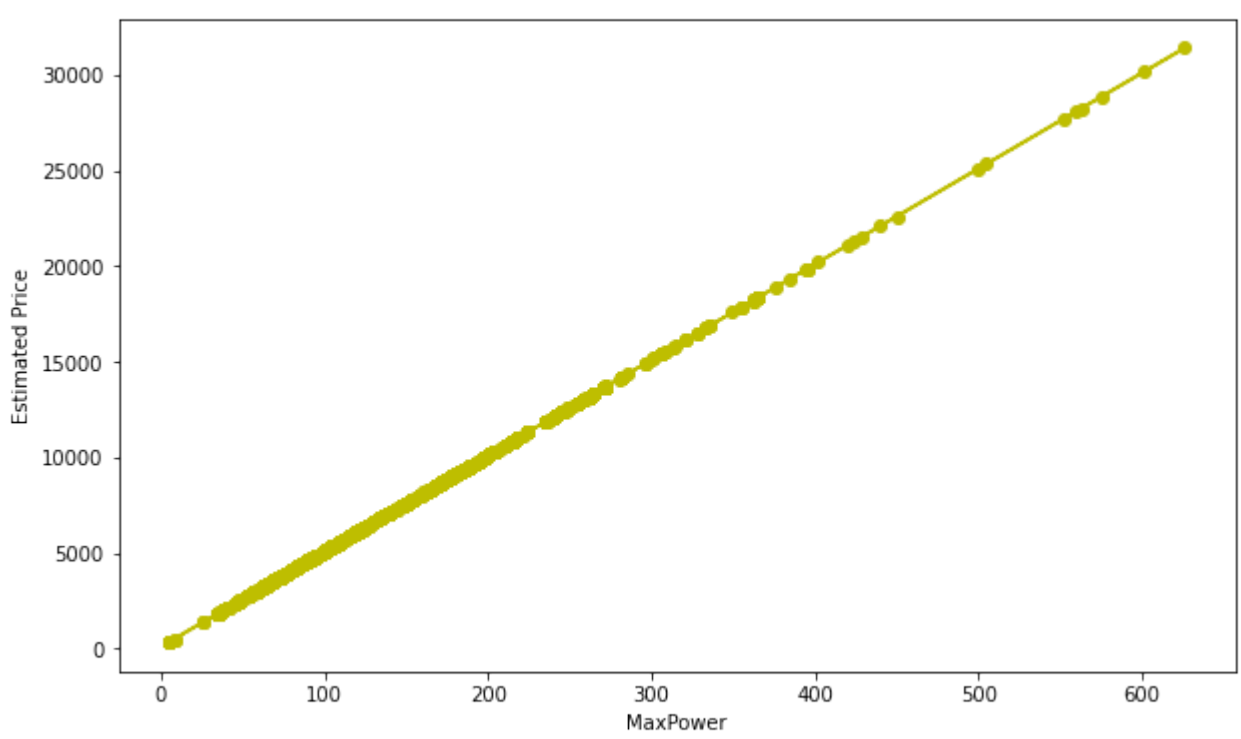
Let's *guess* the values for w and b and use them to estimate the value for price.

```
w = 50
b = 100
```

```
maxpower = data.max_power
estimated_charges = estimate_charges(maxpower, w, b)
```

We can plot the estimated charges using a line graph.

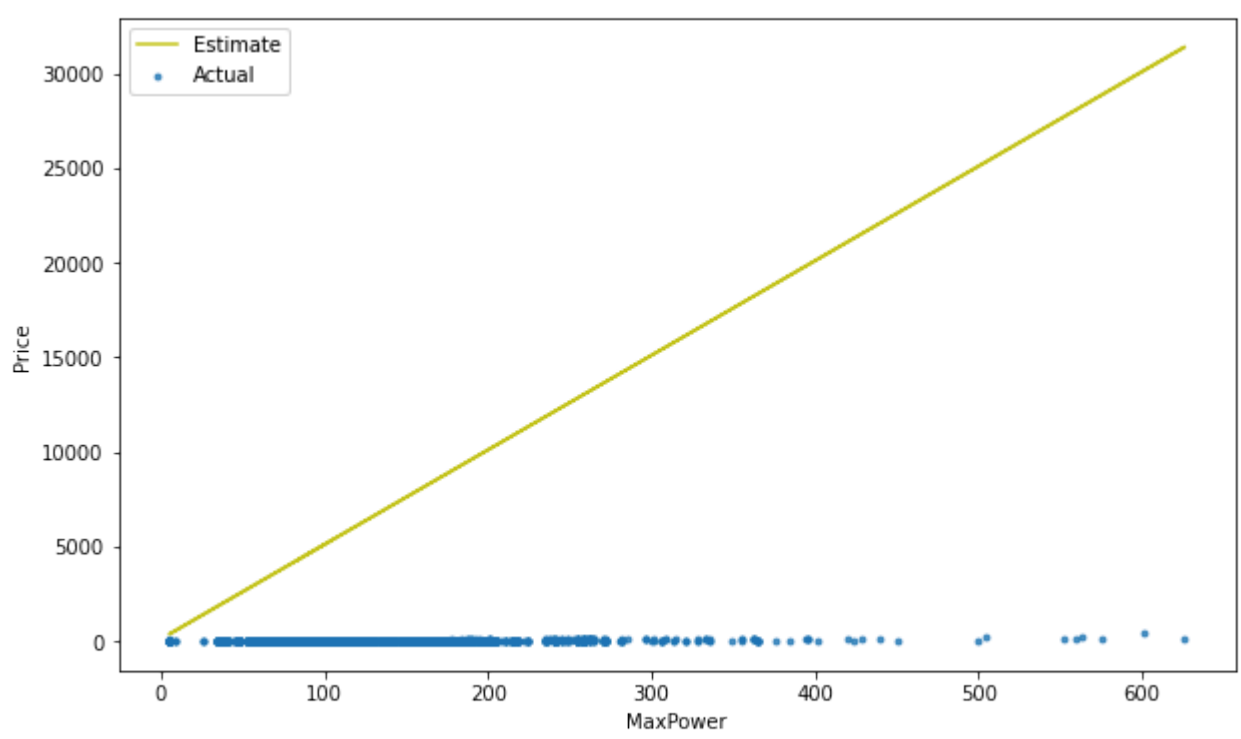
```
plt.figure(figsize=(10,6))
plt.plot(maxpower, estimated_charges, 'y-o');
plt.xlabel('MaxPower');
plt.ylabel('Estimated Price');
```



As expected, the points lie on a straight line.

We can overlay this line on the actual data, so see how well our *model* fits the *data*.

```
target = data.selling_price
plt.figure(figsize=(10,6))
plt.plot(maxpower, estimated_charges, 'y', alpha=0.9);
plt.scatter(maxpower, target, s=8,alpha=0.8);
plt.xlabel('MaxPower');
plt.ylabel('Price')
plt.legend(['Estimate', 'Actual']);
```

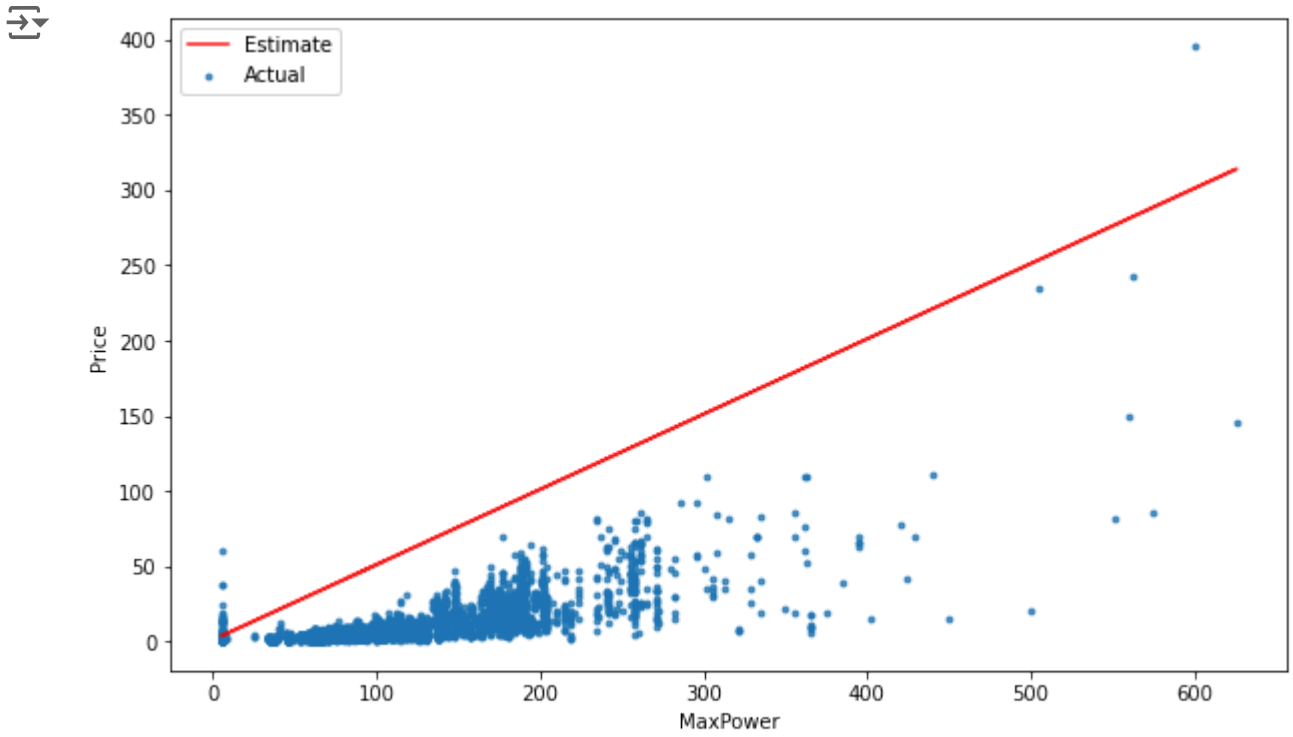


Clearly, the our estimates are quite poor and the line does not "fit" the data. However, we can try different values of w and b to move the line around. Let's define a helper function `try_parameters` which takes `w` and `b` as inputs and creates the above plot.

```
def try_parameters(w, b):
    maxpower = data.max_power
    target = data.selling_price

    estimated_charges = estimate_charges(maxpower, w, b)
    plt.figure(figsize=(10,6))
    plt.plot(maxpower, estimated_charges, 'r', alpha=0.9);
    plt.scatter(maxpower, target, s=8,alpha=0.8);
    plt.xlabel('MaxPower');
    plt.ylabel('Price')
    plt.legend(['Estimate', 'Actual']);
```

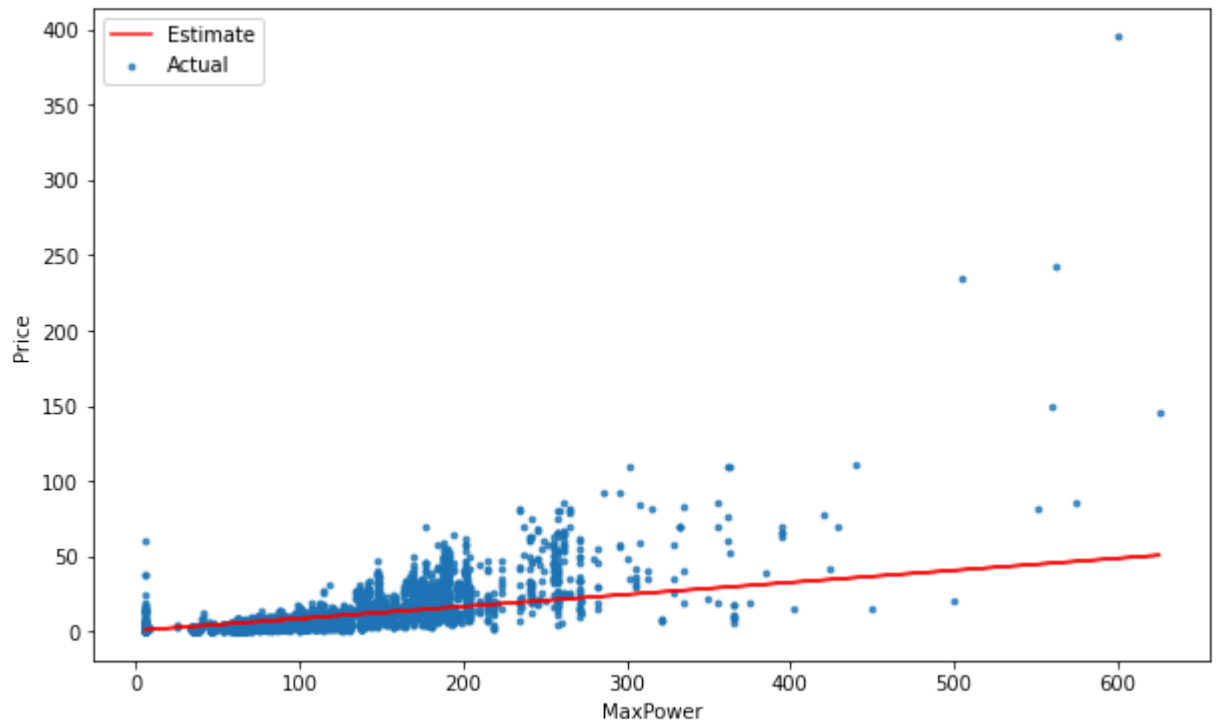
```
try_parameters(0.5, 1)
```



Observation:

- By trying different parameters we see line here is moving towards our points.

```
try_parameters(0.08, 0.5)
```



Observation:

As we change the values, of w and b manually, trying to move the line visually closer to the points, we are *learning* the approximate relationship between "max_power" and "price".

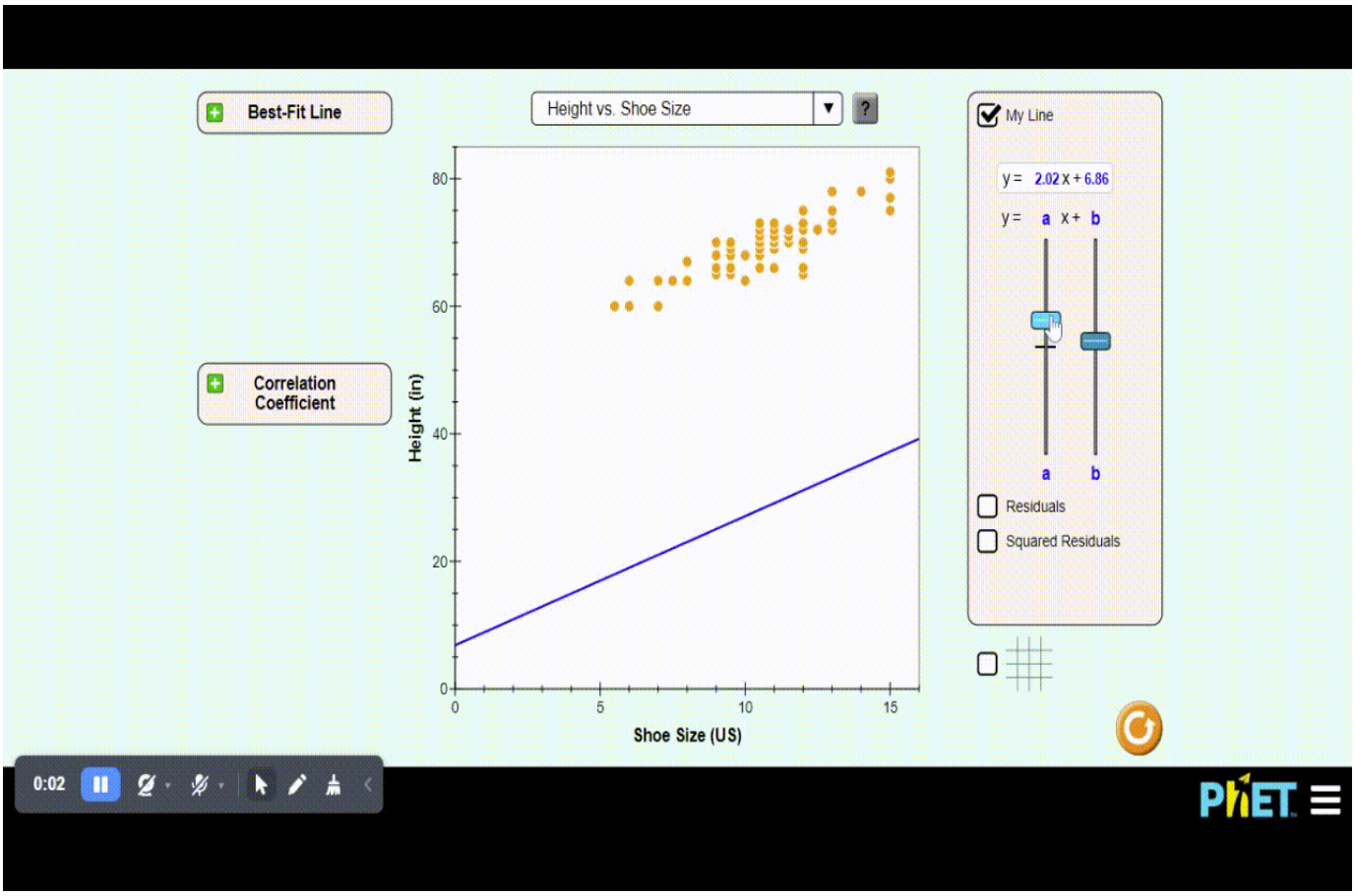
Wouldn't it be nice if a computer could try several different values of `w` and `b` and *learn* the relationship between "max_power" and "price"? To do this, we need to solve a couple of problems:

1. We need a way to measure numerically how well the line fits the points.
2. Once the "measure of fit" has been computed, we need a way to modify `w` and `b` to improve the the fit.

If we can solve the above problems, it should be possible for a computer to determine `w` and `b` for the best fit line, starting from a random guess.

Interactive Visualisation of w and b

- Each time we are changing these parameters and line is adjusting itself.



- Click on the link to try yourself: https://phet.colorado.edu/sims/html/least-squares-regression/latest/least-squares-regression_en.html

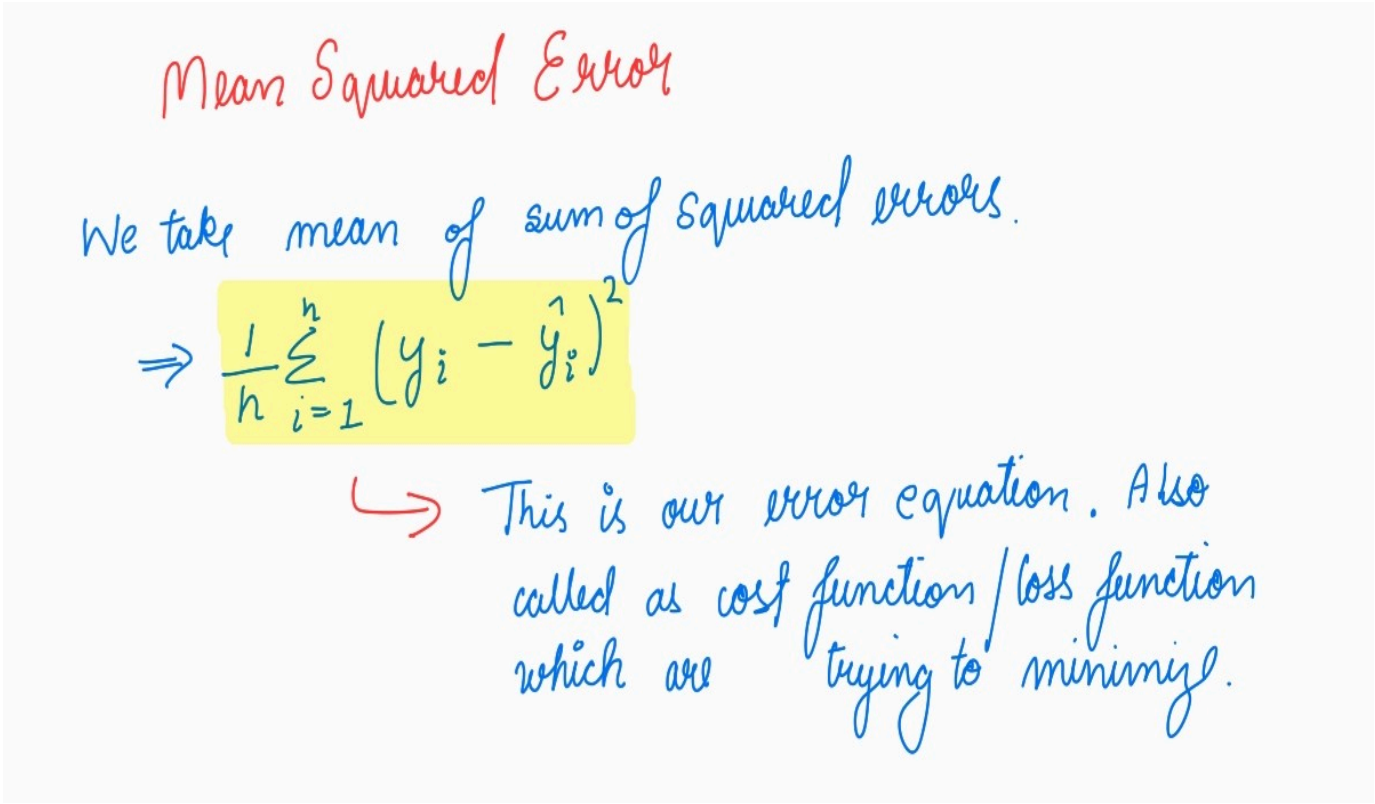
Optimization

- It is a method that computer use to find optimal values of w and b.
- Its uses Gradient descent on top of your cost function to do so.
- Let's see step by step how it works.

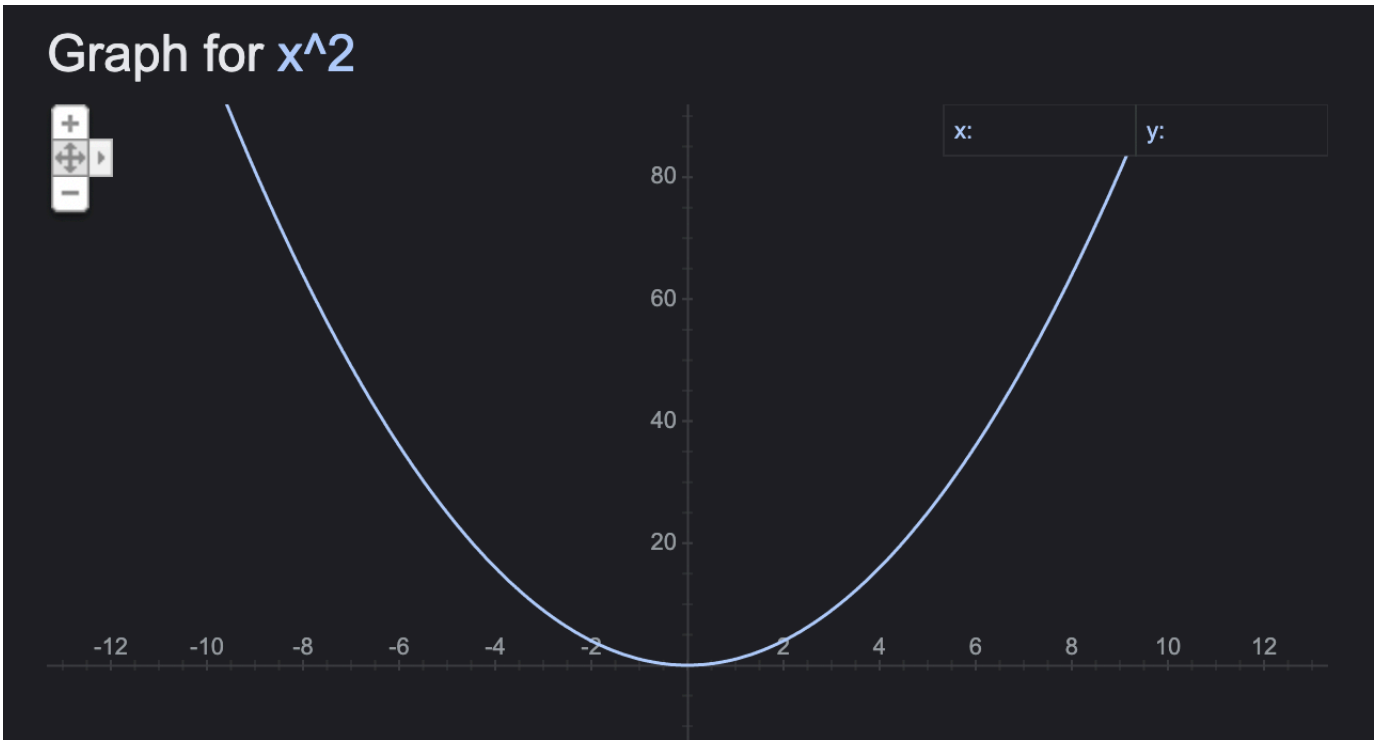
Cost Function

- It is a function that measures the performance of a Machine Learning model for given data.
- Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number.
- We use MSE as our Cost function in linear regression

Mean Squared Error

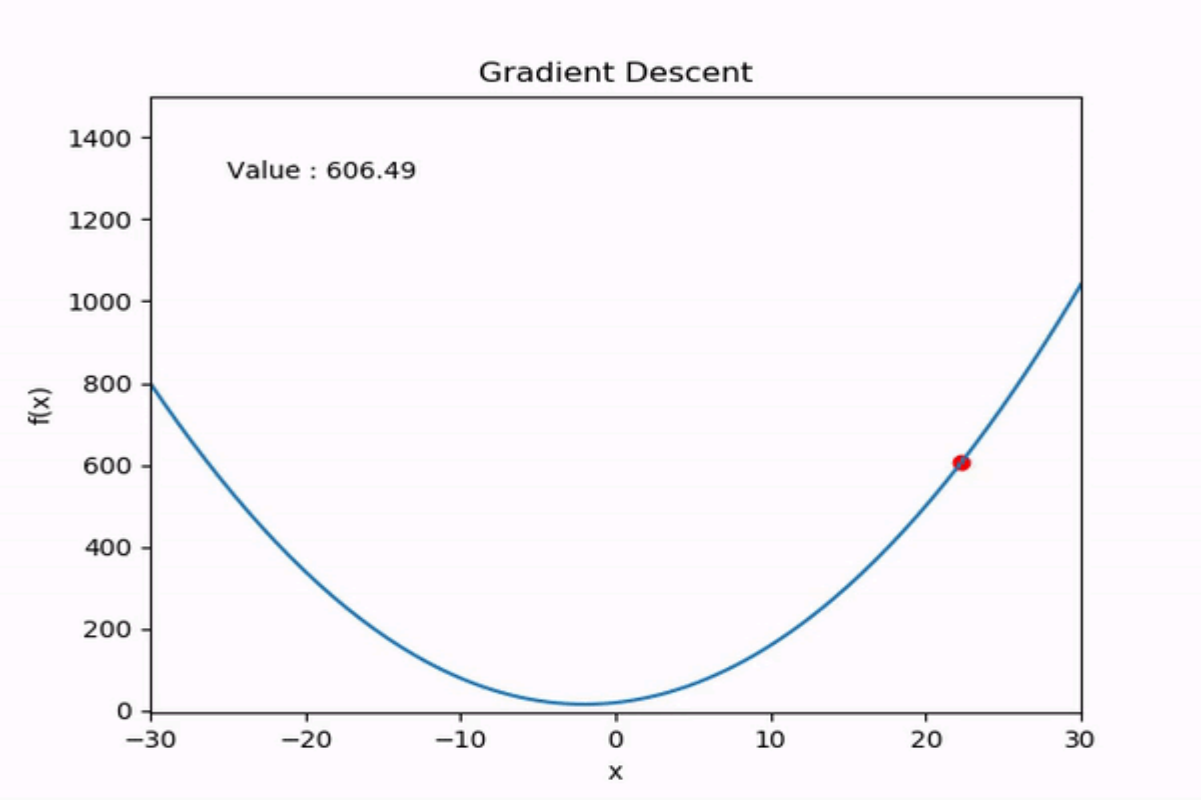


- Here, the error equation is also referred to as loss function/cost function.
- We are trying to minimize our cost function i.e our mean sum of squared error.



Gradient Descent

- We minimized the error by trial and error above — just trying lots of values and visually inspecting the resulting graph.
- There must be a better way? Queue gradient descent.



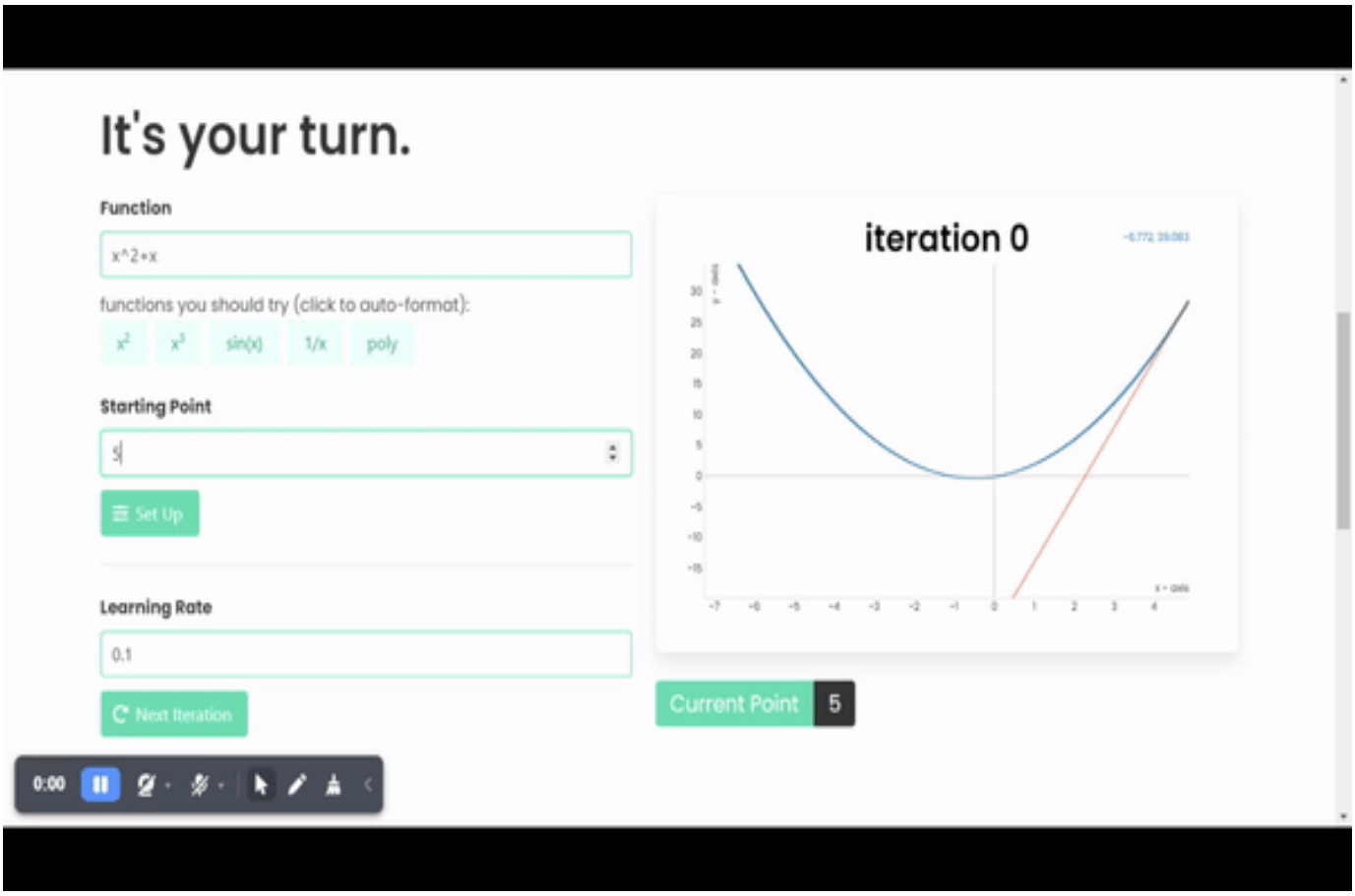
- Gradient Descent is a general function for minimizing a function, in this case the Mean Squared Error cost function.
- **Gradient Descent** basically just does what we were doing by hand — change the parameters, bit by bit, until we hopefully arrived a minimum.

Movement of ball from right to left from a curve slope in first quadrant

- Consider we have the equation of line y=mx+c then,
- The algorithm starts with some value of m and c (usually starts with m=0, c=0). We calculate MSE (cost) at point m=0, c=0.
- Let say the MSE (cost) at m=10, c=10 is 100. Then we change the value of m and c by some amount (Learning Step).
- We will notice a decrease in MSE (cost). We will continue doing the same until our loss function is a very small value or ideally 0 (which means 0 error or 100% accuracy).

Interactive visualisation Gradient descent

- It will show how in steps Gradient descent happens
  - Select a random starting point
  - Set your step size/learning rate
  - Click on next interation to see the magic!



- Click on the link to try it yourself:https://uclaacm.github.io/gradient-descent-visualiser/#playground
- Any Learner who wants to deep dive in maths for Gradient Descent can go through PostRead.

Linear Regression-2

---

7. Code for Univariate LR using Scikit Learn

- In this case, the x axis shows "model" and the y axis shows "selling price of a car". Thus, we're assume the following relationship between the two:

$$price = w \times maxpower + c$$

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

!gdown 1UpLnYA48Vy_lGUMMLG-uQE1gf_Je12Lh
```

```
Downloading...
From: https://drive.google.com/uc?id=1UpLnYA48Vy_lGUMMLG-uQE1gf_Je12Lh
```

```
To: /content/cars24-car-price-clean.csv
100% 7.10M/7.10M [00:00<00:00, 35.0MB/s]
```

```
df = pd.read_csv('cars24-car-price-clean.csv')
df.head()
```

	selling_price	year	km_driven	mileage	engine	max_power	age	
0	-1.111046	-0.801317	1.195828	0.045745	-1.310754	-1.157780	0.801317	-0.43
1	-0.223944	0.450030	-0.737872	-0.140402	-0.537456	-0.360203	-0.450030	-0.32
2	-0.915058	-1.426990	0.035608	-0.582501	-0.537456	-0.404885	1.426990	-0.32
3	-0.892365	-0.801317	-0.409143	0.329620	-0.921213	-0.693085	0.801317	-0.43

```
# define X and y
X = df["model"].values
Y = df["selling_price"].values
```

Scikit-learn

- We have a library which does it for us and provides key insights such as weights of the model, performance of the model.

INSTRUCTOR NOTES

- Link For linear Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

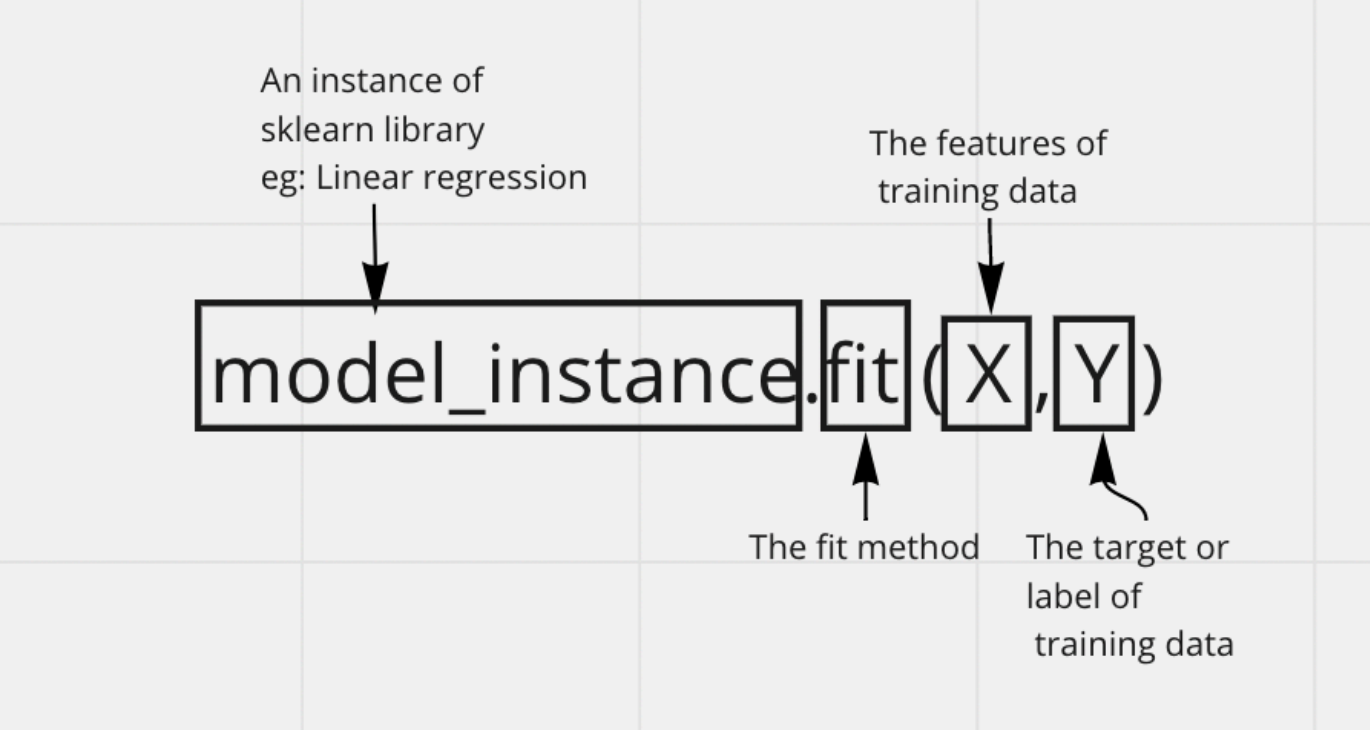
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
type(model)
```

```
sklearn.linear_model._base.LinearRegression
```

- Notice how `model` is just an object of Linear Regression Class

How to train the model using LinearRegression class?

- Scikit-learn provides an inbulit fit method to train the model



- The 'fit' method trains the algorithm on the training data, after the model is initialized. That's really all it does.
- So the sklearn fit method uses the training data as an input to train the machine learning model.

```
model.fit(X,Y)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-20-86f6eb5833af> in <module>
----> 1 model.fit(X,Y)

                                3 frames
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py in
check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,
force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features,
estimator)
    771                     "Reshape your data either using array.reshape(-1, 1) if
"
    772                     "your data has a single feature or array.reshape(1, -1)
"
--> 773                     "if it contains a single sample.".format(array)
    774                 )
    775

ValueError: Expected 2D array, got 1D array instead:
arrav=[-1.12568266 -0.3332271  -0.78980745 ... -0.4486842   0.32802721
```

- But when we used fit method on our X and Y, it gave a shape error

Why was there shape error when using fit method?

- the documentation for fit method says:
  - X (array-like, sparse matrix) of shape (n_samples,n_features)
  - Y (array-like, sparse matrix) of shape (n_samples,) or (n_samples, n_features)

Lets check our X and Y:

```
print(X.shape,Y.shape)
```

```
(19820,) (19820,)
```

- Notice how X and Y shapes are incompatible with the Scikit learn fit method

reshaping X, Y to the required shape

```
X = X.reshape(X.size, 1)
Y = Y.reshape(Y.size, 1)

print(X.shape,Y.shape)
```

```
(19820, 1) (19820, 1)
```

Since now our Data is in required shape

- Now lets fit the model

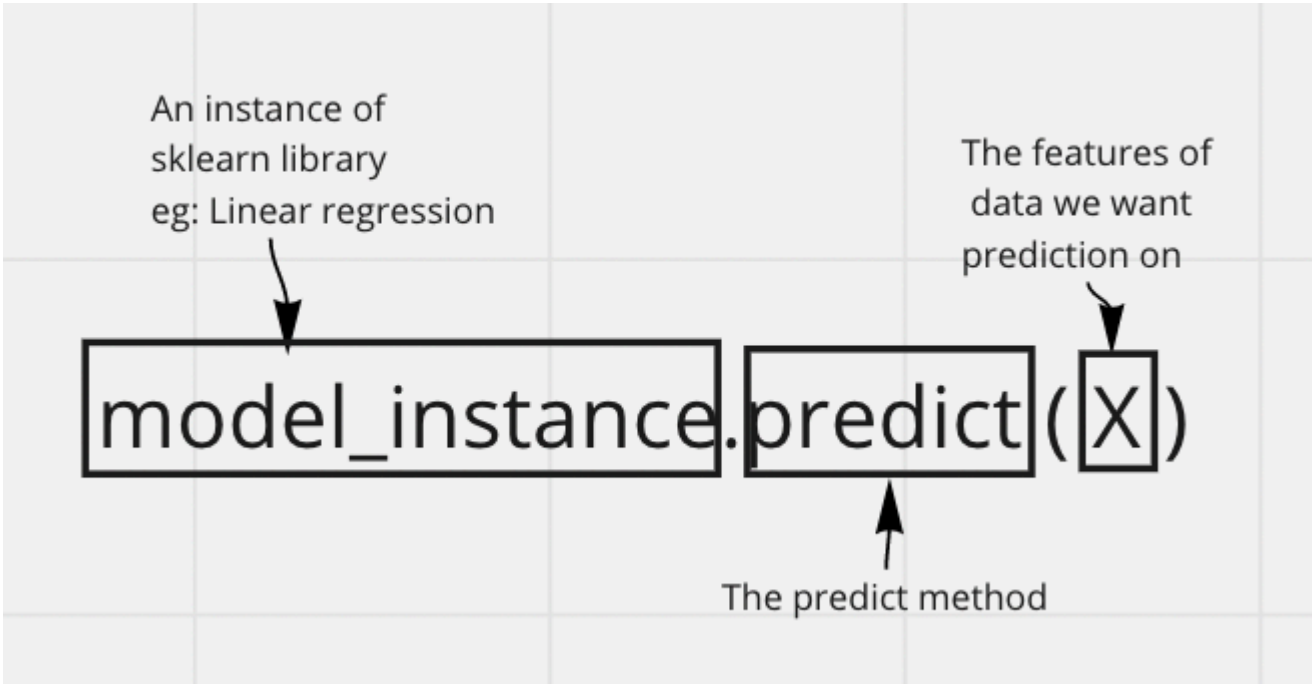
```
model.fit(X,Y)
```

```
LinearRegression()
```

Now that our model is trained,

How do we see the predictions of the trained **model** object?

model.predict() : given a trained model, predict the label of a new set of data. This method accepts one argument, the new data X_new (e.g. model.predict(X_new)), and returns the learned label for each object in the array




```
y_hat = model.predict(X)

print('Predicted')
print(y_hat[:3])
print('Actual')
print(Y[:3])
```

```
Predicted
[[-1.08634131]
 [-0.32158118]
 [-0.76220457]]
Actual
[[-1.11104589]
 [-0.22394353]
 [-0.91505816]]
```

What are the learnt parameters?

```
w0 = model.intercept_
w1 = model.coef_
print(w0, w1)
```

```
[-1.18731936e-16] [[0.96505112]]
```

- Notice how scikit-learn did all the implementation much quickly and with less code.

- This is our final eqn that our LR model has learnt from training the data.

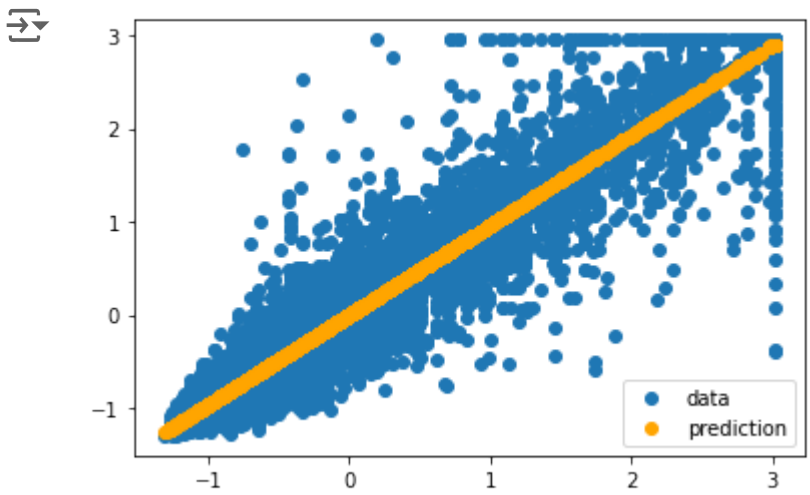
  $sellingprice = w_1 \times model + w_0$

where, w_1 =0.965 and w_0 = -1.18

- w1 is your coefficient of variable "model" and w0 is the intercept.

Lets see how our LR model performed

```
fig = plt.figure()
plt.scatter(X,Y,label='data')
plt.scatter(X,y_hat,color='orange',label='prediction')
plt.legend()
plt.show()
```



Now that is trained how do we check it's performane with a metric

---

8. Metric: Coefficient of Determination (R^2)

- we can check how "close" your prediction is against the real value.

What metrics can we use to check the "closeness"? What metrics have we learnt so far which can do that?

==> MSE

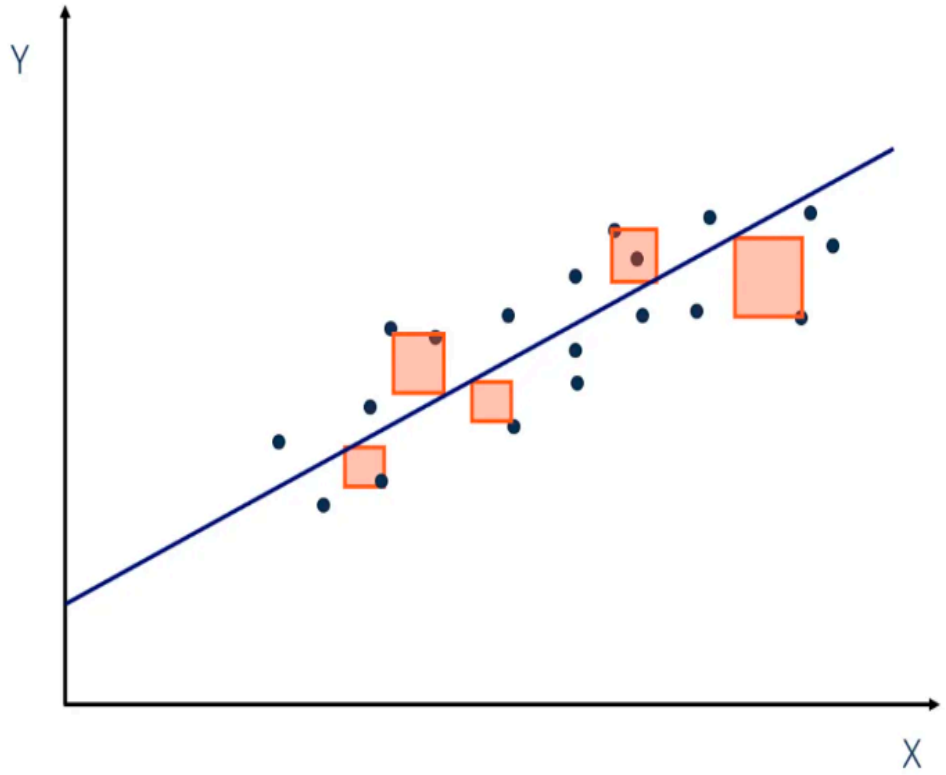
But how small of MSE is small enough? As we have seen, the problem with MSE is that it is not bounded (doesn't have a range).

- **Easier to assess relatively - error of 10K for Alto is bad, but in respect to what? is the error of 10k similarly bad for a Ferrari. If an Alto costs 2 lacs arror of 10 k is 5%, but if the ferrari costs 2 crore error of 10k is 0.05%**

Lets say if my score is 3, 3/5 is good, but 3/10 is bad. Let's try to work on developing such kind of metric

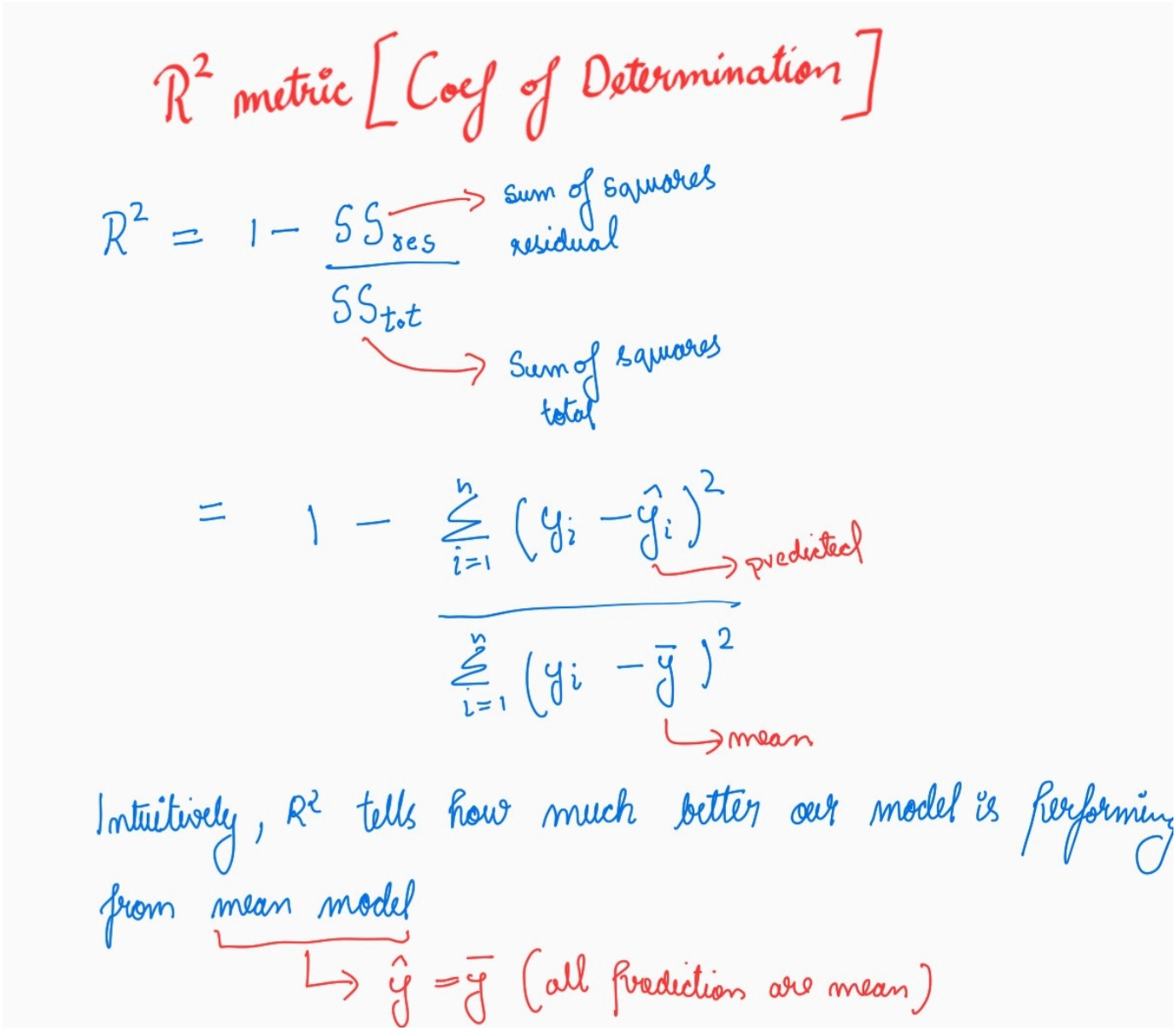
SS(sum of squares total)

- Sum of squares (SS) is a statistical tool that is used to identify the dispersion of data as well as how well the data can fit the model in regression analysis. The sum of squares got its name because it is calculated by finding the sum of the squared differences.



- The total sum of squares is a variation of the values of a dependent variable from the sample mean of the dependent variable. Essentially, the total sum of squares quantifies the total variation in a sample.
- You can use the following steps to calculate the sum of squares:
    - Gather all the data points.
    - Determine the mean/average
    - Subtract the mean/average from each individual data point.
    - Square each total from Step 3.
    - Add up the figures from Step 4.

Sum of Squares Regression (SSR) represents the total variation of all the predicted values found on the regression line or plane from the mean value of all the values of response variables.

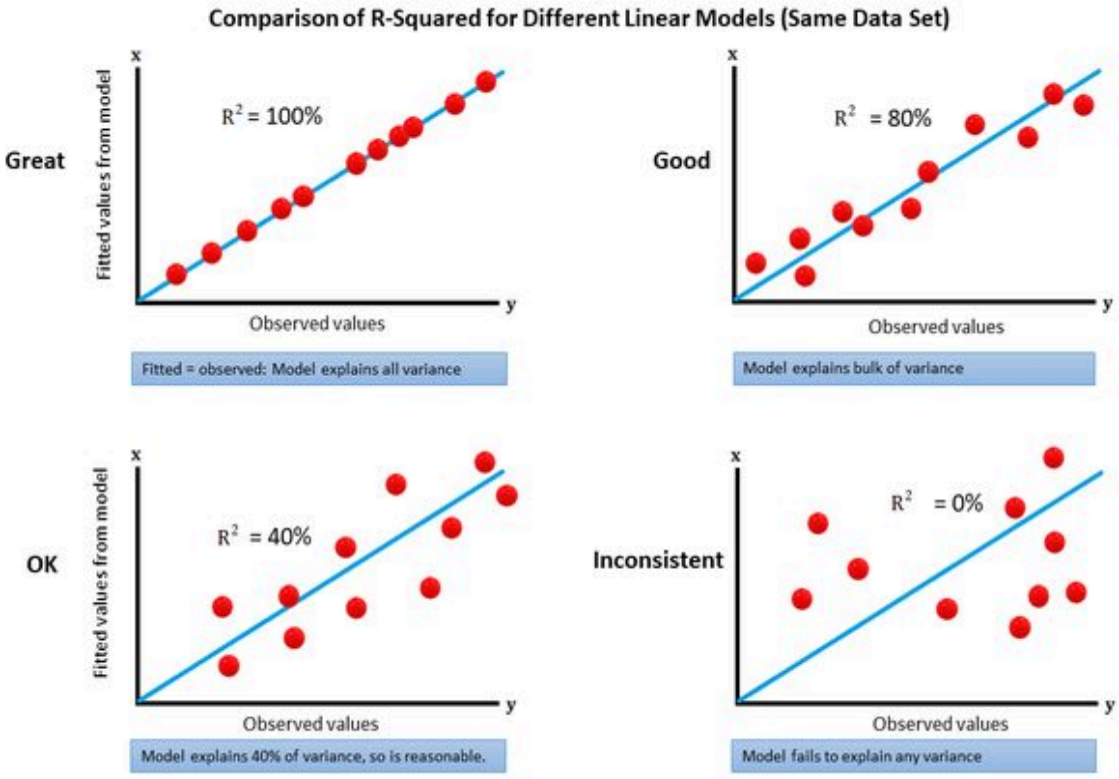


- SS_residual = sum of squared errors for the model that we built
- SS_total = sum of squared errors if the model was mean model.
- Mean model returns mean of yi's as predicted value everytime.
- Intuitively, R-square tells us that how much our model is performing better as compared to mean model.
- **Intuition behind considering mean model as baseline model?** Simplest model that you can build to predict a value is just take mean of all value and return it.

Another definition is “(total variance explained by model) / total variance.”

Score : R2 (R-Squared) or Coefficient of Determination

- R2 score = 1; means perfect model
- R2 score = 0; dumb model, just taking an average of training data.
- R2 score = -1; even worse than dumb model? But is that possible? Theoritically yes.



Question: What do think is best value of R-square?

Ans: 1

Question: Now, what do you think is minimum value of R-square ?

Ans: -inf.

When our model is far worse than mean model i.e. value of SS_residual >> SS_total, the value of R-square will tend to -inf.

- Best value of R-square is 1. When SS residual = 0, R-square = 1
- R-square will be 0 when SS residual = SS total i.e. our model is mean model
- Minimum value of R-square will be -inf. When our model is far worse than mean model, R-square will tend to -inf. (SS residual >> SS total)
- Example when R-square will be large -ve value
    - Imagine you have all horizontal points and you predict the line of best fit as vertical.

How do we get the performance metric for our model using sklearn?

`score` Returns the coefficient of determination of the prediction (R^2)

```
model.score(X,Y)
```

```
0.9313236629576508
```

Does R^2 = 0.93, suggests our model good?

Ans: Yes, it means that our Univariate LR model performed much better than a mean model

But can we be sure by just One Feature, we can estimate the selling price of a Car?

- No, hence we will now look into more features, and how to perform Linear Regression on them

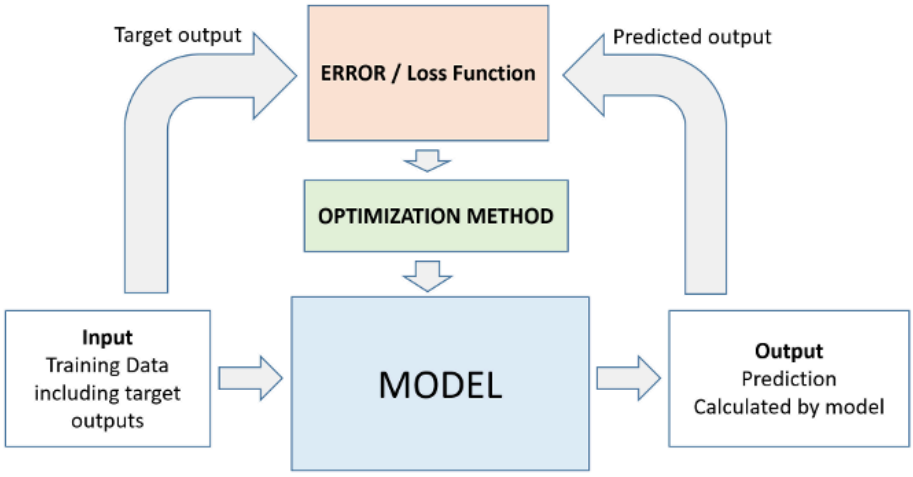
Machine Learning

Congratulations, you've just trained your first Linear Regression model! Machine learning is simply the process of computing the best parameters to model the relationship between some feature and targets.

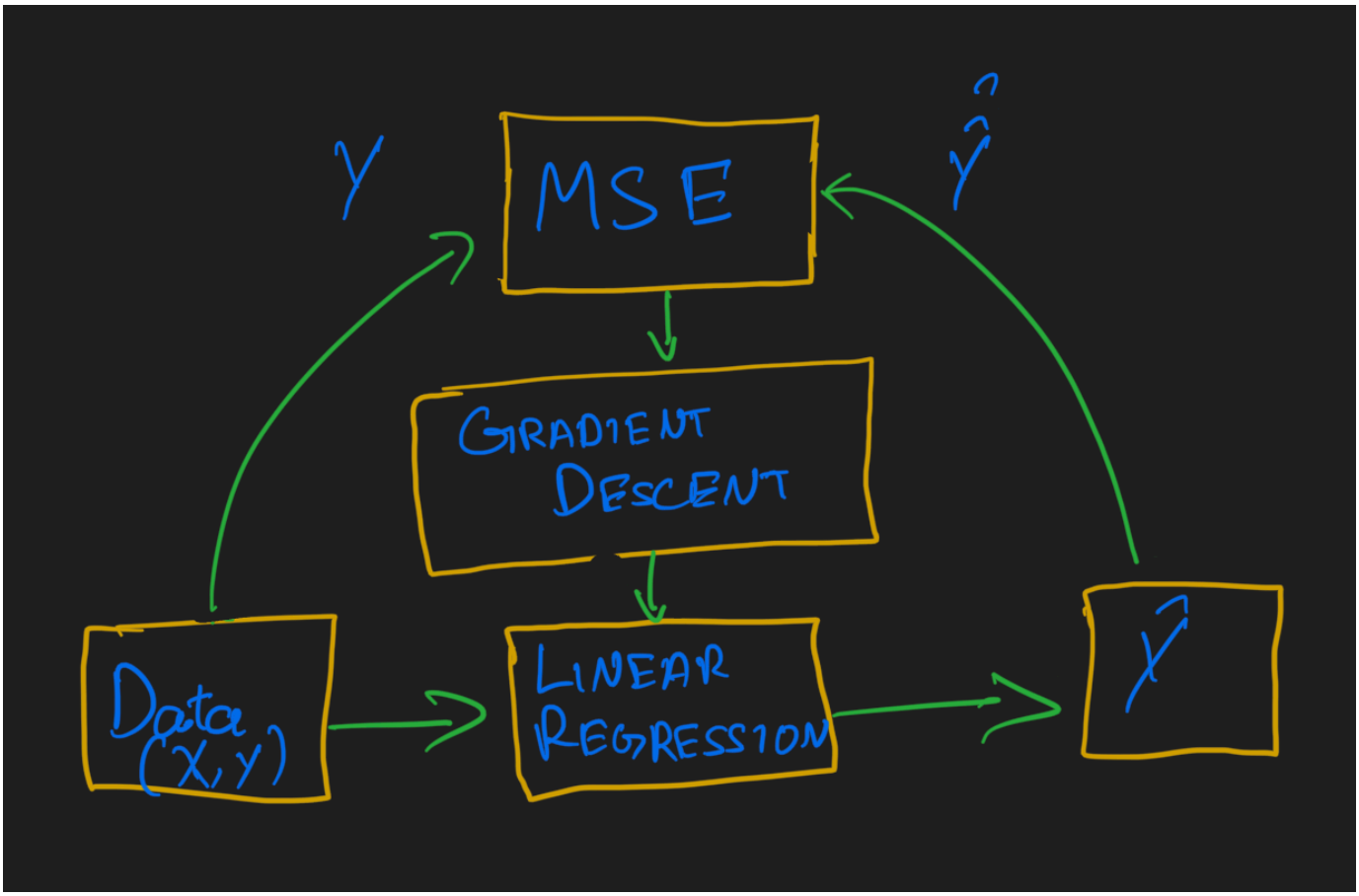
Every machine learning problem has three components:

1. **Model**
2. **Error/Cost Function**
3. **Optimizer**

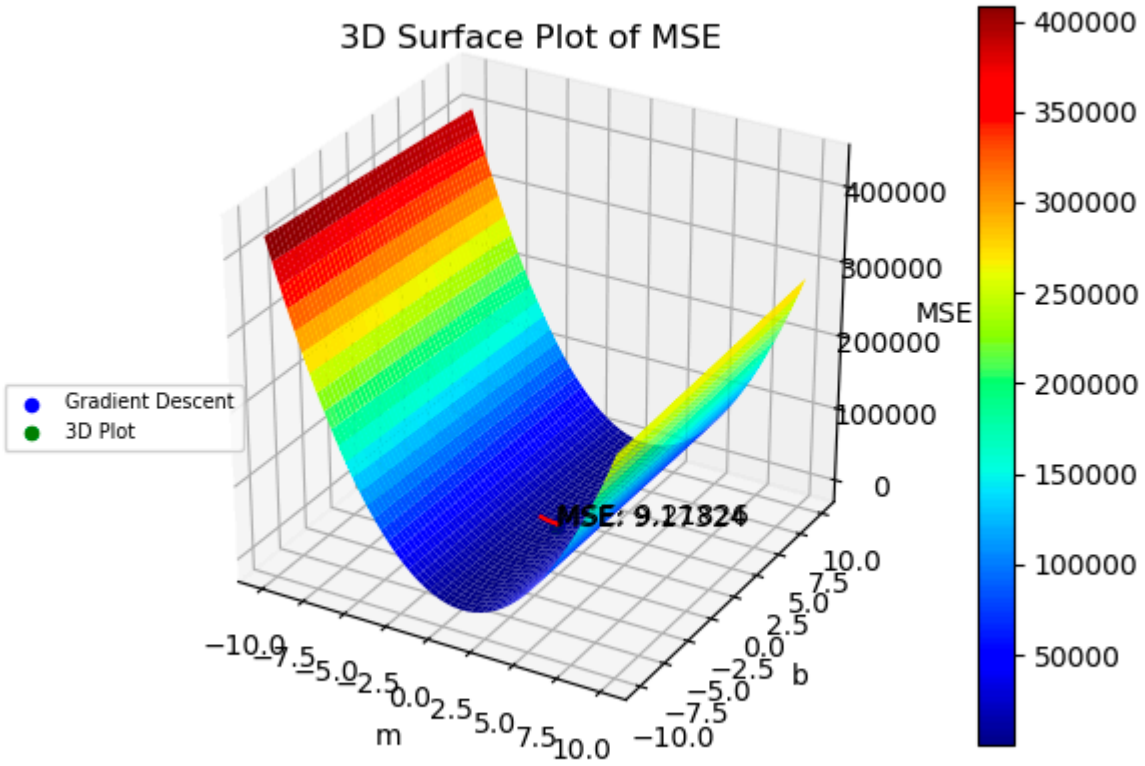
Here's how the relationship between these three components can be visualized:



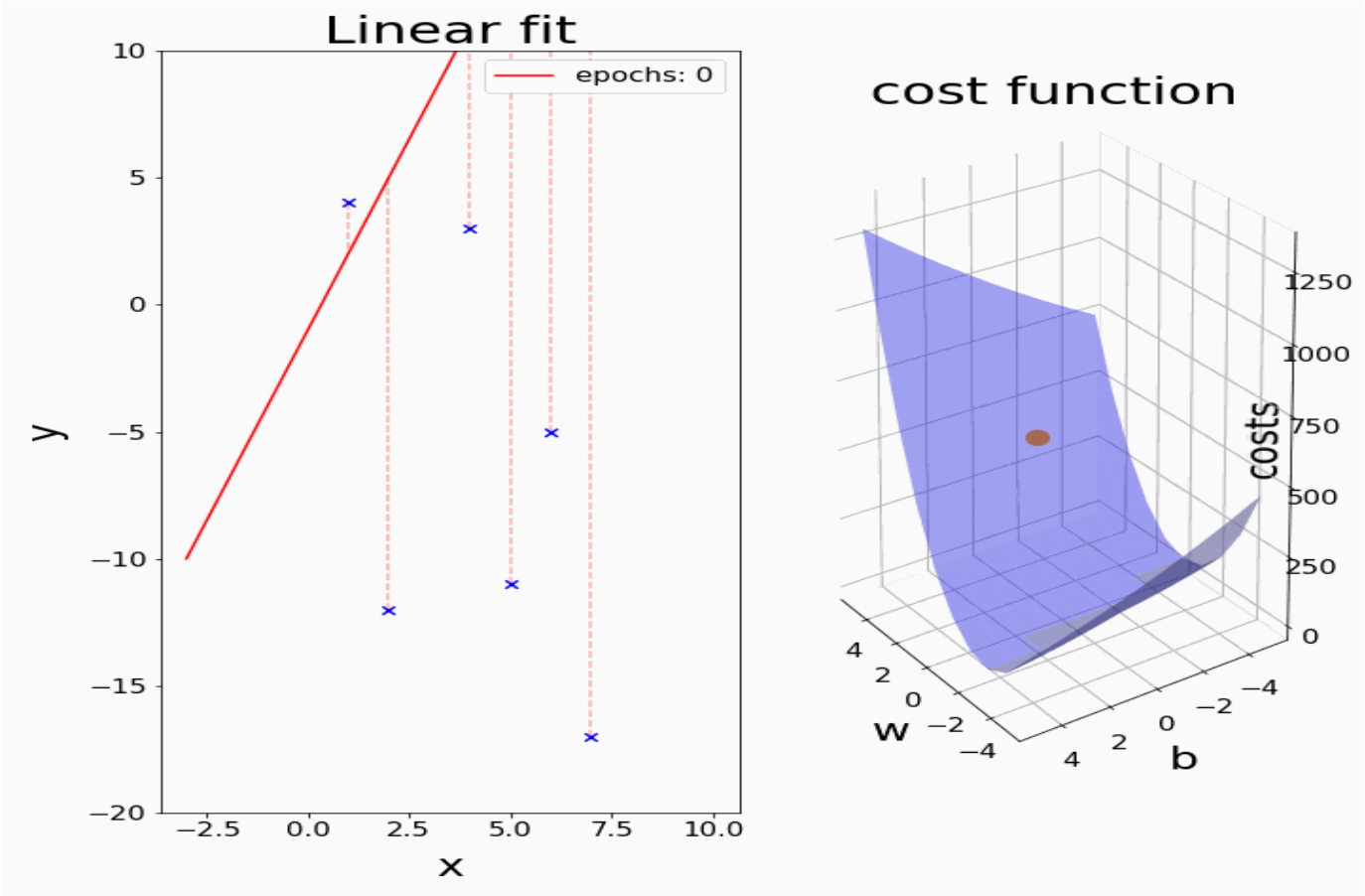
- Your training data with it's y labels is given to model, so that model understands patterns.
- It compares the errors between actual y and predicted y using loss function.
- Then it minimizes the our error using optimization method(GD)



How would MSE look in 3D



Fitting of a line and Gradient descent together with 3D.



- We observe when our w and b changes how our cost fucntion is reducing.
- We can observe the how line comes to the best fit, when GD finds minimum value of cost function.

---

9. Multivariate Linear Regression

Code for Multivariate LR using sklearn

Lets consider all of the features of Cars24 data to predict the selling price

```
df.head()
```

	selling_price	year	km_driven	mileage	engine	max_power	age	
0	-1.111046	-0.801317	1.195828	0.045745	-1.310754	-1.157780	0.801317	-0.43
1	-0.223944	0.450030	-0.737872	-0.140402	-0.537456	-0.360203	-0.450030	-0.32
2	-0.915058	-1.426990	0.035608	-0.582501	-0.537456	-0.404885	1.426990	-0.32
3	-0.892365	-0.801317	-0.409143	0.329620	-0.921213	-0.693085	0.801317	-0.43

- In this case, we are using multiple features of cars and the y axis shows "selling price of a car". Thus, we're assume the following relationship between the features and price of car:

$$price = w_1 \times maxpower + w_2 \times year + w_3 \times engine + \ldots\ldots + c$$

Getting X and Y

- Removing the selling price and taking rest of the features in X

```
X = df[df.columns.drop('selling_price')]
Y = df["selling_price"]
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
cols = X.columns
X[cols] = sc.fit_transform(X[cols])
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:3678: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  self[col] = igetitem(value, i)
```

Convert X and Y from dataframes to numpy array but Why numpy array?

- Since Y is of series datatype before reshaping it to 2d format we convert to numpy array because rehaping is not possible in series type.

```
X = X.to_numpy()
Y = Y.to_numpy()
```

```
print(X.shape, Y.shape)
```

```
(19820, 17) (19820,)
```

```
Y = Y.reshape(Y.size, 1)

print(X.shape,Y.shape)
```

```
(19820, 17) (19820, 1)
```

```
model.fit(X,Y)
```

```
LinearRegression()
```

```
y_hat = model.predict(X)

print('Predicted')
print(y_hat[:3])
print('Actual')
print(Y[:3])
```

```
Predicted
[[-1.18307059]
 [-0.29426139]
 [-0.89454215]]
Actual
[[-1.11104589]
 [-0.22394353]
 [-0.91505816]]
```

```
model.score(X,Y)
```

```
0.942188902908119
```

```
print(model.coef_)
print(model.intercept_)
```

```
[[-4.96486379e+10 -1.82418823e-02 -4.86815023e-02  3.13906593e-02
   2.73634634e-02 -4.96486379e+10  6.07857256e-02  7.94156945e-01
  -1.69523399e-02 -3.47465435e-03  1.40831991e-02  1.27530752e-02
   2.82949517e-03 -2.25842030e-02 -1.29368142e-02 -2.46388004e-02
  -3.26559557e-02]]
[-3.59538383e-06]
```

- Final eqn would be:

$$price = w_1 \times maxpower + w_2 \times year + w_3 \times engine + \ldots\ldots + c$$

where,

c = -3.59538383e-06

w1= -4.96486379e+10, w2= 1.82418823e-02 and so on.

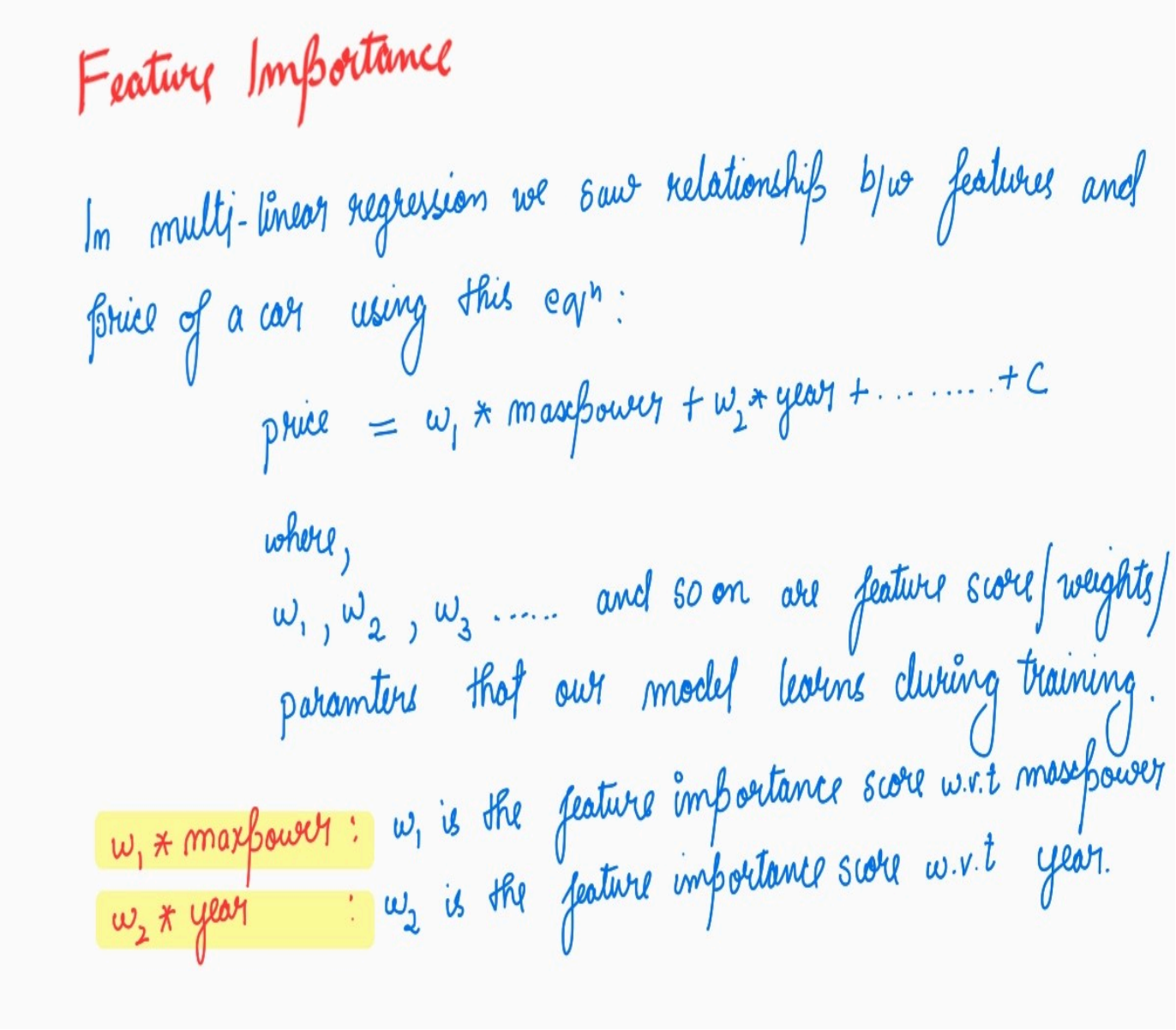
Using all features we get a better r2 score.

Linear Regression - 3

11. Model Interpretibility and Feature Importance

Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. The scores are useful and can be used in a range of situations in a predictive modeling problem, such as:

- Better understanding the data.
- Better understanding a model.
- Reducing the number of input features.



- The sign of a coefficient tells you whether there is a positive or negative correlation between each independent variable and the dependent variable.
- A positive coefficient indicates that as the value of the independent variable increases, the mean of the dependent variable also tends to increase.
- A negative coefficient suggests that as the independent variable increases, the dependent variable tends to decrease.
- The coefficient value signifies how much the mean of the dependent variable changes given a one-unit shift in the independent variable while holding other variables in the model constant. This property of holding the other variables constant is crucial because it allows you to assess the effect of each variable in isolation from the others.

from the equation $price = w_1 * maxpower + w_2 * year + \ldots\ldots + C$

- if the value of w1 is 2 then it means that if all other value being constant one increase in w1 would give an increment of 2x in price

- if the value of w2 is -2 then it means that if all other value being constant one increase in w2 would give an decrement of 2x in price

- **Feature importance scores can provide insight into the dataset**. The relative scores can highlight which features may be most relevant to the target, and the converse, which features are the least relevant. This may be interpreted by a domain expert and could be used as the basis for gathering more or different data.
- **Feature importance can be used to improve a predictive model**. This can be achieved by using the importance scores to select those features to delete (lowest scores) or those features to keep (highest scores). This is a type of feature selection and can simplify the problem that is being modeled, speed up the modeling process, and in some cases, improve the performance of the model.

  Feature importance can be used to improve a buisness decisions. Understanding how the buisness is being affected by the different features can help in making crucial decisions

Now that you are working at cars24 assume you found out that age was the most important feature in the prediction of the car price and luggage space was the least important feature, you can use this information to refine your data, or make age of the car an mandatory information to fill, and luggage space as optional.

if some fetures are not improving the accuracy of the model, you can look into it if the data has some meaning or can be removed

as an buisness devision you can recommend that newer cars are easier to sell or to buy cars that are new

What are the learnt parameters?

- These are the coefficients for all variables we can get the value for each by printing ther model.coef_

```
print(model.coef_)
```

```
[[-4.96486379e+10 -1.82418823e-02 -4.86815023e-02  3.13906593e-02
   2.73634634e-02 -4.96486379e+10  6.07857256e-02  7.94156945e-01
  -1.69523399e-02 -3.47465435e-03  1.40831991e-02  1.27530752e-02
   2.82949517e-03 -2.25842030e-02 -1.29368142e-02 -2.46388004e-02
  -3.26559557e-02]]
```

- if the values are very high we can see that those will effect the prediction most
- by looking at the signs of the coef we can also know that increasing that feature will increase the prediction or decrease the prediction

ols implimenatation with stats model

as we saw earlier OLS method, we have to choose the values of w_1 and w_0 such that, the total sum of squares of the difference between the calculated and observed values of y, is minimised.

- Sklearn also uses ols as the linear regression method, but implimenting it with statsmodel gives us a lot more depth into understanding the model.

https://medium.com/analytics-vidhya/application-and-interpretation-with-ols-statsmodels-499c69ef6834

An intercept is not included by default in the statsmodel implimentation of ols and should be added by the user.

if you do not add the constant the model assumes the interrcept is zero and the line passes through the origin, which may not be the case. the sm.add_constant function returns original array with a constant (column of ones) as the first or last column.

```
# Statmodels implementation of Linear regression
import statsmodels.api as sm
# Lets scale the data, standardization
from sklearn.preprocessing import StandardScaler

X = df[df.columns.drop('selling_price')]
Y = df["selling_price"]

sc = StandardScaler()
cols = X.columns
X[cols] = sc.fit_transform(X[cols])

X_sm = sm.add_constant(X)  #Statmodels default is without intercept, to add intercept we

sm_model = sm.OLS(Y, X_sm).fit()

print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          selling_price   R-squared:                       0.942
Model:                            OLS   Adj. R-squared:                  0.942
Method:                 Least Squares   F-statistic:                 2.017e+04
Date:                Mon, 19 Sep 2022   Prob (F-statistic):               0.00
Time:                        20:42:37   Log-Likelihood:                125.72
No. Observations:               19820   AIC:                            -217.4
Df Residuals:                   19803   BIC:                            -83.24
Df Model:                          16
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.689e-17      0.002   1.57e-14      1.000      -0.003       0.003
year           7.266e+10    4.4e+11      0.165      0.869    -7.9e+11    9.35e+11
```

```
km_driven           -0.0182      0.002     -9.604      0.000      -0.022      -0.015
mileage             -0.0487      0.003    -16.047      0.000      -0.055      -0.043
engine               0.0314      0.004      7.331      0.000       0.023       0.040
max_power            0.0274      0.004      7.108      0.000       0.020       0.035
age               7.266e+10    4.4e+11      0.165      0.869    -7.9e+11    9.35e+11
make                 0.0608      0.003     19.873      0.000       0.055       0.067
model                0.7941      0.004    192.928      0.000       0.786       0.802
Individual          -0.0170      0.002     -9.509      0.000      -0.020      -0.013
Trustmark Dealer    -0.0035      0.002     -2.020      0.043      -0.007      -0.000
Diesel               0.0141      0.007      1.992      0.046       0.000       0.028
Electric             0.0128      0.002      6.903      0.000       0.009       0.016
LPG                  0.0028      0.002      1.484      0.138      -0.001       0.007
Petrol              -0.0226      0.007     -3.123      0.002      -0.037      -0.008
Manual              -0.0130      0.002     -5.638      0.000      -0.017      -0.008
5                   -0.0246      0.006     -4.150      0.000      -0.036      -0.013
>5                  -0.0327      0.006     -5.239      0.000      -0.045      -0.020
==============================================================================
Omnibus:                     4568.728   Durbin-Watson:                   1.915
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           179651.533
Skew:                           0.335   Prob(JB):                         0.00
Kurtosis:                      17.734   Cond. No.                     7.78e+14
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
[2] The smallest eigenvalue is 1.49e-25. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:3678: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  self[col] = igetitem(value, i)
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning
  x = pd.concat(x[::order], 1)
```

Shivank : Explain why add_constant and what all extra details OLS provide here

const part added above, ols details give as post read

p value and feature importance

each feature has a p value attached to it, and we can use this value to undertand if this feature

significance level is the amount of change a feature will affect towards the final output i.e. how important is this feature and how much it affects the final output. Generally, we take 5%/0.05 significance level by default.

null hypotheseis the feature is not important

alternate hypothesis the feature is important

- p-value refers to the hypothesis of the significance level.

- eg : p-value of 0.994, null hypothesis is true this column does not provide any noticeable change to the output and can be easily removed without consequences.
- a p-value of 0.001, null hypothesis is rejected i.e. is provides very significant change to the output.

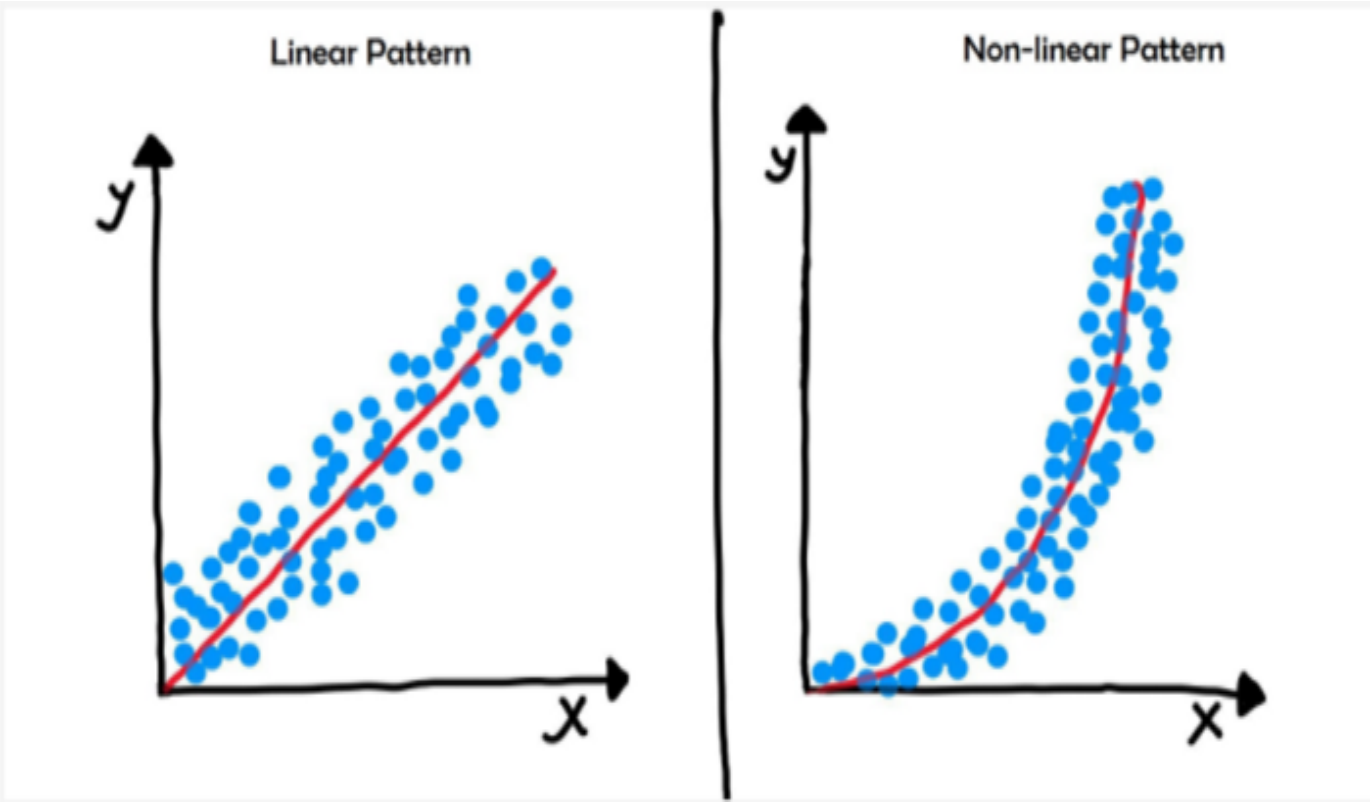
> INSTRUCTOR NOTES notice that we have p value for year and age to be very high becuase of collinearity. we will remove this via vif later and then we can compare and see what has happened.

---

12. Assumptions of Linear Regression

1.Assumption of Linearity

- Linear Regression is that relations between the independent and dependent variables must be linear.
- if your relationships are not linear, you should not use a linear model, but rather a non-linear model



2.Features are not multi-collinear

Multicollinearity is the phenomenon when a number of the explanatory variables are strongly correlated.

Correlation explains how one or more variables are related to each other. Positive Correlation: means that when the value of one variable increase then the value of the other variable(s) also increases. Negative Correlation: means that when the value of one variable increase then the value of the other variable(s) decreases.

So why do we want to have strong correlations between each independent variable and the dependent variable, but no correlation between independent variables? The reason is that if two independent variables are correlated, they explain the same information. The model will not be able to know which of the two variables is actually responsible for a change in the dependent variable.

- two features are collinear when one feature can be expressed as a linear function of the other ie: $f_2 = af_1 + b$
- if there are multiple features that may be collinear, we can say that $f_n = a_1 f_1 + a_2 f_2 + a_3 f_3 + b$
- here we can say that because these can be represented as linear functions of each other, fn, f1,f2,f3 show multicollinearity

assume you have two feature length of car in centimeters and length of car in meters

- these features will be perfectly collinear
- $price = w_0 + w_1 power + w_2 max speed + w_3 length\ cm + w_4 length\ m.....$
- $length\ cm = 100 * length\ m$
- Now it will be the same if we write $price = w_0 + w_1 power + w_2 max speed + w_3 * 100 length\ m + w_4 length\ m.....$

this will confuse the model as to which feature is actually effecting to the target variable

- Now that we understand Multi-Colinearity and its issues

How can we remove these highly correlated features?

VIF (Variance Inflation Factor)

" VIF determines the strength of the correlation between the independent variables. It is predicted by taking a variable and regressing it against every other variable. "

- Variance inflation factor (VIF) is used to detect the severity of multicollinearity in the ordinary least square (OLS) regression analysis.

- Multicollinearity inflates the variance . It makes the coefficient of a variable consistent but unreliable.
- VIF measures the number of inflated variances caused by multicollinearity.

VIF score of an independent variable represents how well the variable is explained by other independent variables.

$$VIF_i = \frac{1}{1 - R_i^2}$$

where, R2i is the R2-value obtained by regressing the ith predictor on the remaining predictors.

Assume 4 independent variables X1, X2, X3, and X4.

- Regress X1 with (X2, X3, and X4) to get its R-Squared (R^2) statistic. VIF of X1 = 1 / (1 – R^2).
- Likewise, regress X2 with (X1, X3, and X4); X3 with (X1, X2, and X4); X4 with (X1, X2, and X3) to get their respective R-Squared and VIFs.

- It can be infered that if $VIF \Uparrow$, that feature is more collinear
- Rule of thumb:
  - if $VIF_j > 10$: Very High multiCollinearity
  - if 5 <= $VIF_j < 10$: High multiCollinearity
  - VIF of intercept is to be ignored.
  - VIF of Dummy variable should not be used.

Now lets see if there are any multi-collinear features in Cars24 data

Shivank : More details on VIF is required like how it is calculated. Its not much maths heavy so can explained well.

added above

```
# VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor


vif = pd.DataFrame()
X_t = X
```

```
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/stats/outliers_influence.py:193: R
  vif = 1. / (1. - r_squared_i)
```

	Features	VIF
0	year	inf
5	age	inf
13	Petrol	17.95
10	Diesel	17.03
16	>5	13.32
15	5	12.08
3	engine	6.27
7	model	5.80
4	max_power	5.08
6	make	3.20
2	mileage	3.15
14	Manual	1.80
12	LPG	1.24
1	km_driven	1.23
11	Electric	1.17
8	Individual	1.09

- VIF values tends to be infinity when there is a perfect correlation between the variables
- Any variable with a VIF of 10 or above is considered strongly correlated with other variables.

We can see that year and age have an inf vif score that means they are perfectly corellated, and we know that older the year more the age and they will vary in alinear fashion itself.

That means we can use either one of the and the results would still be the same.

Now for variables like desiel and petrol we know that those are dummy variables

Taking all the columns which have VIF < 10 and one from features that are collinear to each other

```
cols2 = ['age','engine','model','max_power','make','mileage','Manual','Diesel','Petrol','

X2 = X[cols2]

X2_sm = sm.add_constant(X2)  #Statmodels default is without intercept, to add intercept w

sm_model = sm.OLS(Y, X2_sm).fit()

print(sm_model.summary())
```

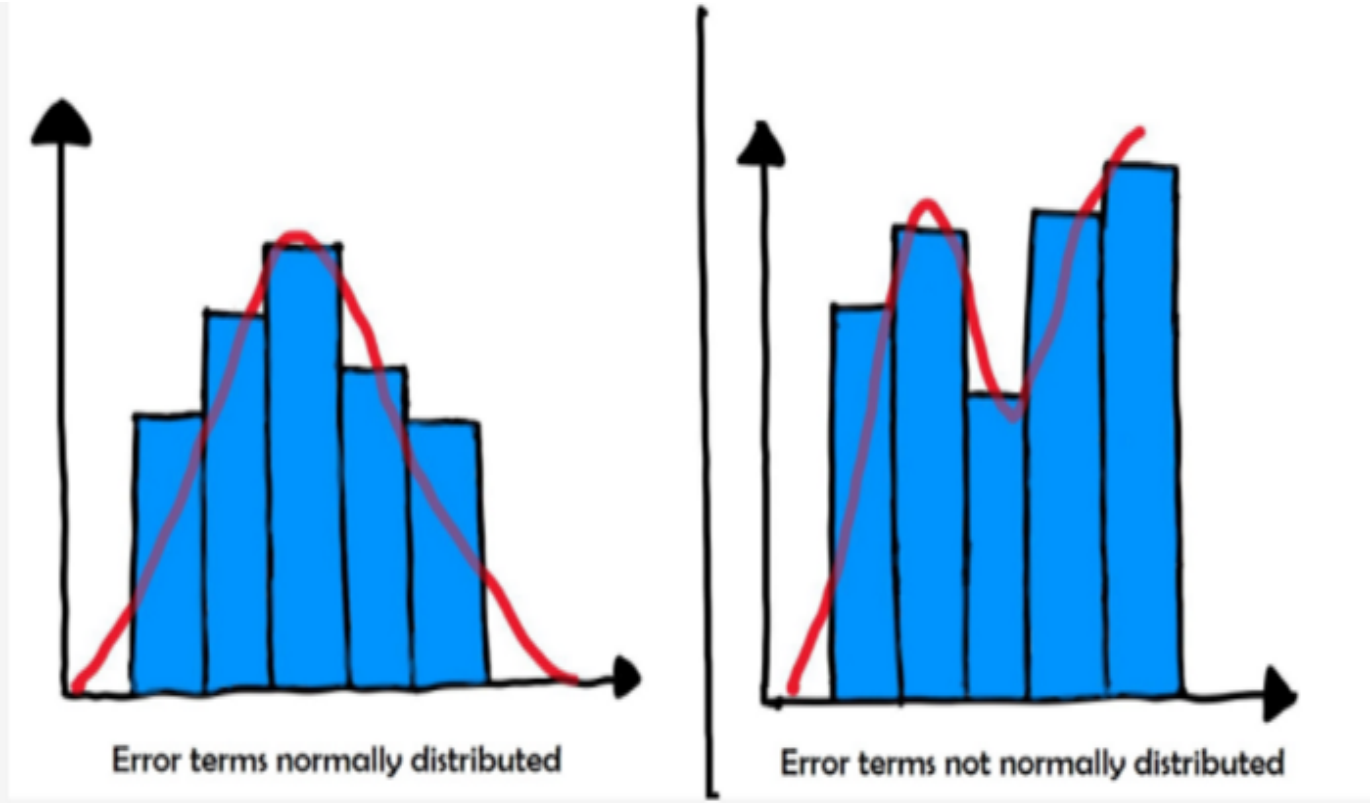
```
                            OLS Regression Results
==============================================================================
Dep. Variable:          selling_price   R-squared:                       0.942
Model:                            OLS   Adj. R-squared:                  0.942
Method:                 Least Squares   F-statistic:                 2.456e+04
Date:                Mon, 19 Sep 2022   Prob (F-statistic):               0.00
Time:                        21:19:58   Log-Likelihood:                 21.855
No. Observations:               19820   AIC:                            -15.71
Df Residuals:                   19806   BIC:                             94.81
Df Model:                          13
Covariance Type:            nonrobust
================================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const             2.689e-17      0.002   1.57e-14      1.000      -0.003       0.003
age                 -0.1230      0.002    -52.865      0.000      -0.128      -0.118
engine               0.0559      0.004     14.884      0.000       0.049       0.063
model                0.8057      0.004    200.418      0.000       0.798       0.814
max_power            0.0206      0.004      5.535      0.000       0.013       0.028
make                 0.0623      0.003     20.386      0.000       0.056       0.068
mileage             -0.0223      0.002     -9.330      0.000      -0.027      -0.018
Manual              -0.0105      0.002     -4.580      0.000      -0.015      -0.006
LPG                  0.0049      0.002      2.839      0.005       0.002       0.008
km_driven           -0.0140      0.002     -7.460      0.000      -0.018      -0.010
Electric             0.0070      0.002      3.900      0.000       0.004       0.011
Individual          -0.0162      0.002     -9.064      0.000      -0.020      -0.013
Trustmark Dealer    -0.0046      0.002     -2.683      0.007      -0.008      -0.001
5                    0.0008      0.002      0.350      0.726      -0.004       0.005
==============================================================================
Omnibus:                     4663.039   Durbin-Watson:                   1.914
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           183740.891
Skew:                           0.370   Prob(JB):                         0.00
Kurtosis:                      17.898   Cond. No.                         5.68
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning
  x = pd.concat(x[::order], 1)
```

Here we can see that the p value for age is now improved.

3.Errors are normally distributed

- The third assumption of Linear Regression is that the residuals should follow a normal distribution.



- Errors (e_i) are normally distributed.
- e_i = yi - y_i_hat
- If you plot the error distribution and it is not normal, then it means there are outliers present in data.
- It means there are some observations where the error is very large. i.e. likelyhood of outliers is more.

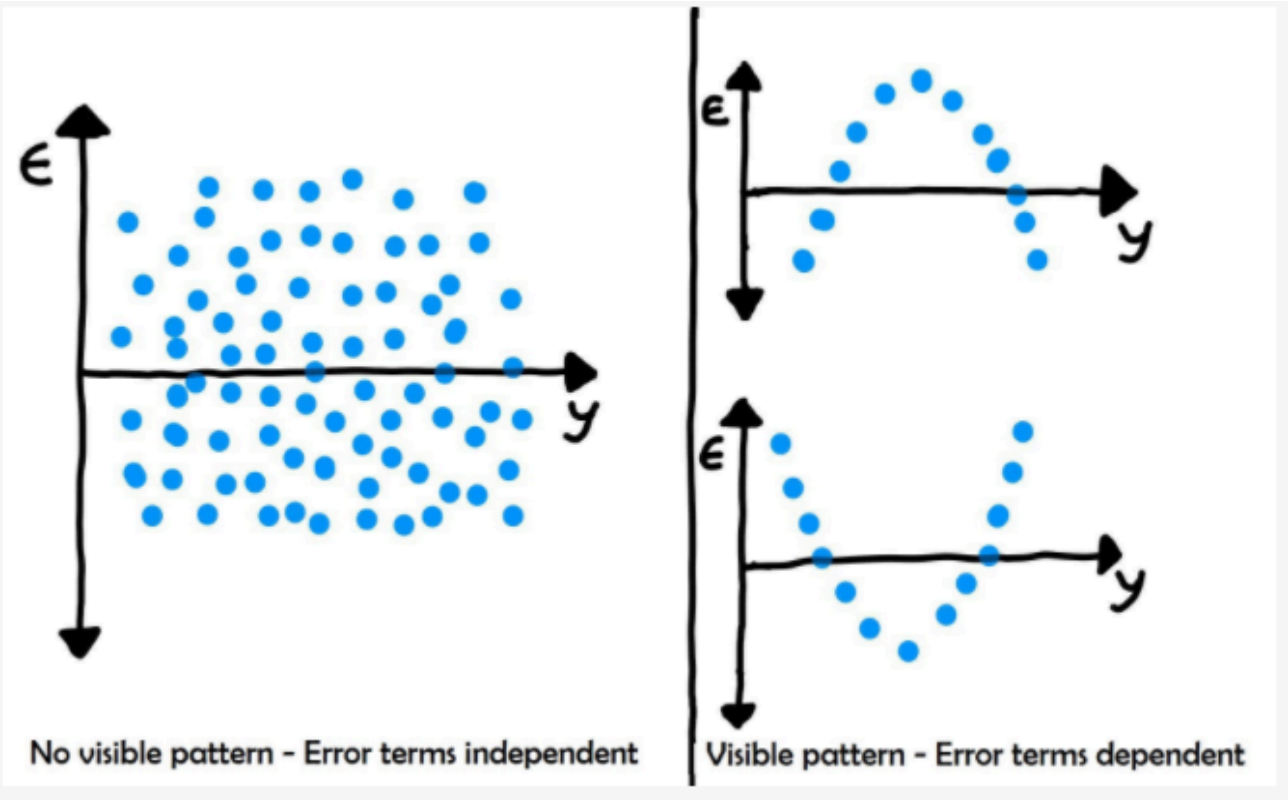
- The calculation of confidence interval and variable significance is based on this assumption. For example, you are trying to analyze which variables are useful to predict car price and you have selected the factors based on a 5% significance level. If the distribution of error significantly deviates from the mean 0 normal distribution, the factors you choose to be significant may not actually be significant enough to contribute to car price changes.

in more simpler terms if we have outier than it would effect our analysis, becuase the errors for them would be large

Shivank : Explain why?

added above

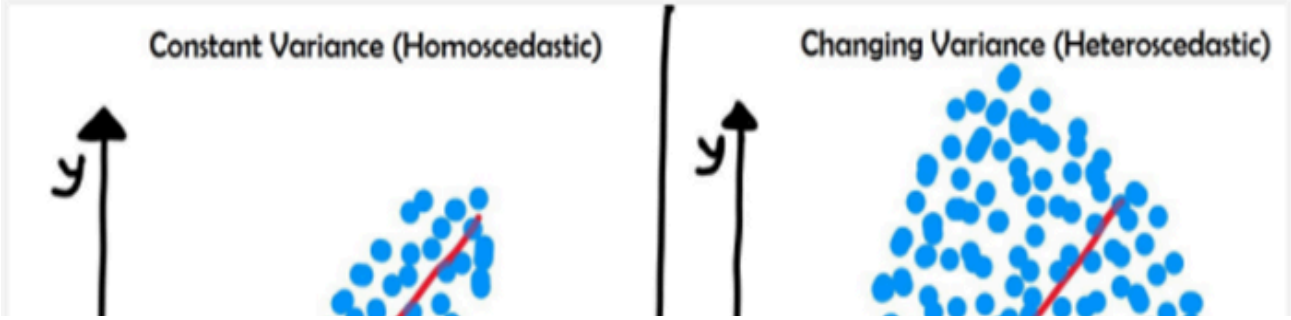
4. Error terms must be independent

- To check this assumption draw a scatter plot between the target variable and the error term. A Scatter plot should not show visible patter.
- 

5. Homoscedasticity: Error term have constant variance.

Homoscedasticity means to be of “The same Variance”, the error terms will be the same and of very little variance.

- A scatterplot of residuals versus predicted values is good way to check for homoscedasticity. You should get a graph like the left graph above.



- Presence of Heteroscedasticity makes the coefficients less precise and hence the correct coefficients are further away from the population value.
- Heteroscedasticity is also likely to produce p-values smaller than the actual values. This is due to the fact that the variance of coefficient estimates has increased but the standard OLS (Ordinary Least Squares) model did not detect it. Therefore the OLS model calculates p-values using an underestimated variance. This can lead us to incorrectly make a conclusion that the regression coefficients are significant when they are actually not significant.

we can check for hetroscedacity by plotting the error vs prediction plot

we can see that the error are not same for all y_i so we can say that there is some hetroscedacity