Multiple Linear Regression

The primary usage of this regression model is when there is a case when multiple independent variables are present, and there is a dependent variable; given there is a linear relation between both the independent and dependent.

Equation:

$$y = b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n$$

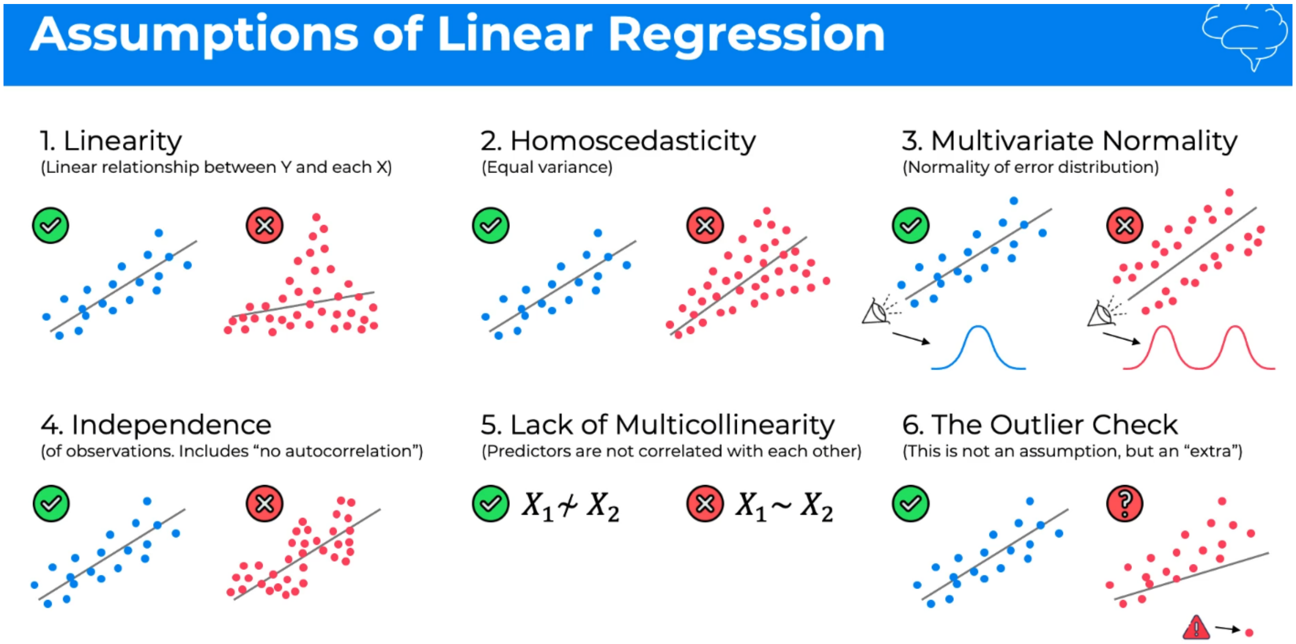
Where:

- `y` is the dependent variable
- b_0 is the y-intercept
- b_1 is the slope coefficient 1
- X_1 is the independent variable 1
- b_2 is the slope coefficient 2
- X_2 is the independent variable 2

Assumptions of Linear Regression:

- Linearity:
  - Linear relationship between Y and each X
- Homoscedasticity
  - Equal variance
- Multivariate Normality
  - Normality of error distribution
- Independence
  - No autocorrelations, meaning previous values affecting next values
- Lack of Multicollinearity
  - Predictors are not correlated with each other
- Outlier Check
  - Are the outliers in the dataset significantly affecting our linear regression line, if yes

consider another model



Understanding P-Value:

- Statistical Significance
  We have to understand the situation, we can assume we are in a completely fair situation or a completely unfair environment. This is denoted as
  H_0 for null hypothesis (fair)
  H_1 for not null hypothesis (unfair)
  We initially assume that any outcome is in fair environment, and that particular outcome has a P-Value of less than 100%. Lets look at a coin-flip example, if we get the following results:

```
Roll     Outcome     P Value
  1       Tails        0.5
  2       Tails        0.25
  3       Tails        0.12
  4       Tails        0.06
  5       Tails        0.03
  6       Tails        0.01
```

The probability of flipping tails 6 consecutive times is ~1%, this represents our `p-value`. Because the `p-value` is so low, we raise alarms and start questioning if this indeed is a null hypothesis. In statistics, we have a confidence threshold, $\alpha = 0.05$, if the `p-value` is below this threshold, assume something is not fair and reject that hypothesis. This confidence threshold can be adjustable depending on experiment and usage.

Example:

Given 50 companies, with their costs of operations (R&D, Administration, Marketing), their location (statewide) and their Profits, determine which companies are worth investing in, and how does each independent variable (R&D/Administration/Location/Marketing) affect the profits. Should an investor prioritize one over the other?

- So how does this equation look like?
  - $$y = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + ???$$
  - where X_1 is the amount spent on R&D
  - where X_2 is the amount spent on Administration
  - where X_3 is the amount spent on Marketing
    The question is, what is `???`, it represents the states of each company, but how do we add that in our regression equation? `State` is a categorical variable, and therefore we must create dummy variables.
    For each category (lets say we have two states; Ontario and Quebec), we must create additional columns. For each company, if they are located in Ontario, it has a value of 1 for the `ontario` column and a value of 0 for the `quebec` column.

  Now the equation looks like:

  $$y = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + b_4D_4$$

  - where D_4 is the dummy variable for Ontario. Do not add all the dummy variables, as essentially you are choosing either Ontario or Quebec. Another reason for this is as follows, when we have 2 dummy variables, D_1 and D_2, we have the following relation

    $$D_2 = 1 + D_1$$

  - This means, D_2 is dependent on D_1 and this violates one of the assumptions for linear regression models, `Multicollinearity`.
  - One rule to remember, for each categorical data, always have 1 less dummy variable to avoid this trap.

Building a Model:

We have many independent variables for our dependent variable in a multiple linear regression model. At many times, we have to remove some of the predictors from our dataset. The reasoning is, if we have nonsense data going into our model, the model will use that nonsense data to give us predictions which make no sense. Keep the important the variables.

5 methods for building models

- All In:
  - Basically use all the data regardless of quantity and quality. We should not use this method unless we have prior knowledge of this model, or we have to given some requirements. Or we are preparing for backward elimination.
- Backward Elimination:
  - Select the significance level to stay in the model => SL = 0.05 (5%)
  - Fit the complete model with all possible predictors
  - Consider the predictor with the highest `P-Value`, if `P-Value > SL` move to step 4; otherwise finish.
  - Remove that predictor.
  - Fit the model without this variable.
    Repeat the steps until the highest predictor values are less than the significance level. After which the model is finished.
- Forward Selection:
  - Select the significance level to enter the model => SL = 0.05 (5%)
  - Fit all simple regression models $y \sim x_n$. Select the one with the lowest `P-Value`.
  - Keep this variable and fit all possible models with an extra predictor added to the ones you have(had)
  - Once again, consider the predictor with the lowest `P-Value`, if `P < SL` for all values in the equation, we are now done with the model. Take the model previous as that is the one that is complete.
- Bidirectional Elimination:
  - Select the significance level to enter and stay in the model: SL_Enter = 0.05 SL_Stay = 0.05.
  - Perform the next step of forward selection, new variables must have `P < SL_ENTER to enter`
  - Perform all steps of Backward Elimination (old variables must have `P < SL_Stay` to stay)
  - Repeat step 2. Until at some points no new variables can enter and old variables cannot leave. At that point the model is finished.
- Score Comparison:
  - Select a criterion of goodness of fit (Akaike Criterion)
  - Construct all possible Regression Models,
  - Select the one with the best criterion
  - Model is ready
    This method is not feasible, lets say 10 columns. This will mean 1023 models.

Applying Multiple Linear Regressions

- **Import Libraries**

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

- **Import Dataset**

```
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

- **Encoding Categorical Data**
  This dataset contains an independent variable called `state` which is categorical and not numerical. We must apply encoding to that independent variable.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])],
remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

Remember to change the index to index of column which you want to apply Encoding.

- **Splitting Dataset into Training and Testing Sets**
  Now we have to split dataset for Training and Testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)
```

- **Training Data on our Linear Regression Model**
  Note, we do not need to identify or specify which method of creating the model, classes such as `sklearn` allow already do this for us, meaning we can count on it to give us the best possible model for us.

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

- **Predicting Data using Model**
  Simple Linear Regressions only had 1 independent and 1 dependent variable, meaning it

was possible for us to graph the predictions and visualize it. Now though, it is not possible as for example we have 4 independent variable and 1 dependent variable, meaning we would need a 5-Dimensional Graph to plot the results.
Instead, we can display two vectors; the first one being the real value in the test dataset (also known as the real data from a testing set or known as a groundtruth) and then the second vector is the predicted values of the same test set.
We can use these two different vectors to compare our results.

```
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

First step is to get the predictions vector by calling the `.predict(X_test)`. For our vectors, we define our precision for our numerical data to two decimal points. Next we print out both the vectors, groundtruth prediction from our test dataset and predicted values from our model for the same test. We use our numpy concatenate function, where we first add our vectors needed to concatenate. Reshape them vertically, and close out the first parameter.
Next parameter is the `axis`, if you have two horizontal vectors and need to concatenate them, you set `axis-> 0` and it will vertically concatenate them. In our case, we have two vertical vectors and we will perform horizontal contactization by setting `axis->1`.

- **To get a prediction based on an input:**

```
print(regressor.predict([[1.0, 0.0, 0.0, 160000, 130000, 300000]]))
```

- **To get equation of our linear regression**

```
print(regressor.coef_)
print(regressor.intercept_)
```