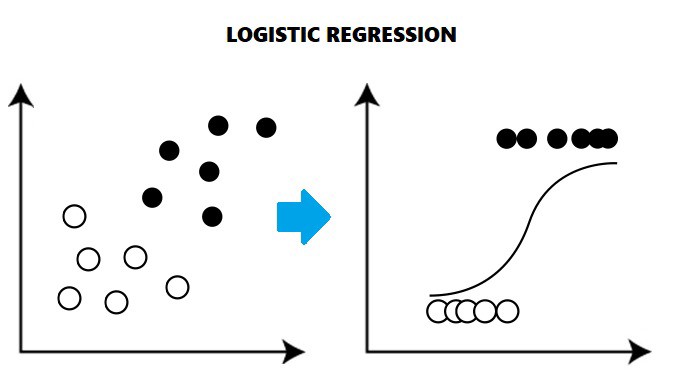
Assignment 13

1. What are the different types of logistic Regression?

Ans:

**Logistic Regression** : Logistic regression is **a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set**. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variable.



Types of Logistic Regression:

There are three main types of logistic regression: binary, multinomial and ordinal. They differ in execution and theory. Binary regression deals with two possible values, essentially: yes or no. Multinomial logistic regression deals with three or more values. And ordinal logistic regression deals with three or more classes in a predetermined order.

### Binary logistic regression

Binary logistic regression was mentioned earlier in the case of classifying an object as an animal or not an animal—it’s an either/or solution. There are just two possible outcome answers. This concept is typically represented as a 0 or a 1 in coding. Examples include:

* Whether or not to lend to a bank customer (outcomes are yes or no).
* Assessing cancer risk (outcomes are high or low).
* Will a team win tomorrow’s game (outcomes are yes or no).

### Multinomial logistic regression

Multinomial logistic regression is a model where there are multiple classes that an item can be classified as. There is a set of three or more predefined classes set up prior to running the model. Examples include:

* Classifying texts into what language they come from.
* Predicting whether a student will go to college, trade school or into the workforce.
* Does your cat prefer wet food, dry food or human food?

### Ordinal logistic regression

Ordinal logistic regression is also a model where there are multiple classes that an item can be classified as; however, in this case an ordering of classes is required. Classes do not need to be proportionate. The distance between each class can vary. Examples include:

* Ranking restaurants on a scale of 0 to 5 stars.
* Predicting the podium results of an Olympic event.
* Assessing a choice of candidates, specifically in places that institute ranked-choice voting

1. What is the difference between the outputs of the Logistic Model and the Logistic function?

Ans:

**Output of Logistic Model:**

The output of a logistic regression model is **the probability of our input belonging to the class labeled with 1**. And the complement of our model's output is the probability of our input belonging to the class labeled with 0

**Output of Logistic Function:**

The output of logistical regression is reported in terms of odds ratios, which is **the numerical odds (bounded by 0 and infinity) of the binary, dependent variable being true, given a one-unit increase in the independent variable**.

1. **How do we handle categorical variables in Logistic Regression?**

**Ans**:

Logistic regression models are a great tool for analysing binary and categorical data, allowing you to **perform a contextual analysis to understand the relationships between the variables, test for differences, estimate effects, make predictions, and plan for future scenarios**

Machine learning models require all input and output variables to be numeric.

This means that if your data contains categorical data, you must encode it to numbers before you can fit and evaluate a model.

The two most popular techniques are an **Ordinal Encoding** and a **One-Hot Encoding**.

**Nominal Variable** (*Categorical*). Variable comprises a finite set of discrete values with no relationship between values.

**Ordinal Variable**. Variable comprises a finite set of discrete values with a ranked ordering between values.

## Encoding Categorical Data

There are three common approaches for converting ordinal and categorical variables to numerical values. They are:

* Ordinal Encoding
* One-Hot Encoding
* Dummy Variable Encoding

Let’s take a closer look at each in turn.

### Ordinal Encoding

In ordinal encoding, each unique category value is assigned an integer value.

For example, “red” is 1, “green” is 2, and “blue” is 3.

This is called an ordinal encoding or an integer encoding and is easily reversible. Often, integer values starting at zero are used.

For some variables, an ordinal encoding may be enough. The integer values have a natural ordered relationship between each other and machine learning algorithms may be able to understand and harness this relationship.It is a natural encoding for ordinal variables. For categorical variables, it imposes an ordinal relationship where no such relationship may exist. This can cause problems and a one-hot encoding may be used instead.

This ordinal encoding transform is available in the scikit-learn Python machine learning library via the [OrdinalEncoder class](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html).

By default, it will assign integers to labels in the order that is observed in the data. If a specific order is desired, it can be specified via the “*categories*” argument as a list with the rank order of all expected labels.

**One-Hot Encoding**

For categorical variables where no ordinal relationship exists, the integer encoding may not be enough, at best, or misleading to the model at worst.

Forcing an ordinal relationship via an ordinal encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

In this case, a one-hot encoding can be applied to the ordinal representation. This is where the integer encoded variable is removed and one new binary variable is added for each unique integer value in the variable.

In the “color” variable example, there are three categories, and, therefore, three binary variables are needed. A “1” value is placed in the binary variable for the color and “0” values for the other colors.

**Dummy Variable Encoding**

The one-hot encoding creates one binary variable for each category.

The problem is that this representation includes redundancy. For example, if we know that [1, 0, 0] represents “*blue*” and [0, 1, 0] represents “*green*” we don’t need another binary variable to represent “*red*“, instead we could use 0 values for both “*blue*” and “*green*” alone, e.g. [0, 0].

This is called a dummy variable encoding, and always represents C categories with C-1 binary variables.

In addition to being slightly less redundant, a dummy variable representation is required for some models.

For example, in the case of a linear regression model (and other regression models that have a bias term), a one hot encoding will case the matrix of input data to become singular, meaning it cannot be inverted and the linear regression coefficients cannot be calculated using [linear algebra](https://machinelearningmastery.com/linear-algebra-machine-learning-7-day-mini-course/). For these types of models a dummy variable encoding must be used instead.

**OrdinalEncoder Transform**

An ordinal encoding involves mapping each unique label to an integer value.

This type of encoding is really only appropriate if there is a known relationship between the categories. This relationship does exist for some of the variables in our dataset, and ideally, this should be harnessed when preparing the data.

In this case, we will ignore any possible existing ordinal relationship and assume all variables are categorical. It can still be helpful to use an ordinal encoding, at least as a point of reference with other encoding schemes.

We can use the [OrdinalEncoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html) from scikit-learn to encode each variable to integers. This is a flexible class and does allow the order of the categories to be specified as arguments if any such order is known.

**OneHotEncoder Transform**

A one-hot encoding is appropriate for categorical data where no relationship exists between categories.

The scikit-learn library provides the [OneHotEncoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html) class to automatically one hot encode one or more variables.

By default the *OneHotEncoder* will output data with a [sparse representation](https://machinelearningmastery.com/sparse-matrices-for-machine-learning/), which is efficient given that most values are 0 in the encoded representation. We will disable this feature by setting the “*sparse*” argument to *False* so that we can review the effect of the encoding.

Once defined, we can call the *fit\_transform()* function and pass it to our dataset to create a quantile transformed version of our dataset.

1. **What are the assumptions made in Logistic Regression?**

Ans:

Assumptions:

1. In a logistic regression , there is No or minimum muliticollinearity amongst independent variable(X). This means that the independent variables should not be too highly correlated with each other.
2. In logistic regression , relationship between ln=∑ᶱixi  must be linear .
3. Logistic regression includes independence of errors.
4. The error terms (residuals) do not need to be normally distributed.
5. Homoscedasticity is not required.
6. The dependent variable in logistic regression is not measured on an interval or ratio scale.
7. logistic regression assumes linearity of independent variables and log odds.  although this analysis does not require the dependent and independent variables to be related linearly, it requires that the independent variables are linearly related to the log odds.
8. Logistic regression typically requires a large sample size.  A general guideline is that you need at minimum of 10 cases with the least frequent outcome for each independent variable in your model. For example, if you have 5 independent variables and the expected probability of your least frequent outcome is .10, then you would need a minimum sample size of 500 (10\*5 / .10).
9. Why Can’t we use MSE as a cost function for Logistic Regression?

Ans:

In the logistic regression we are using sigmoid function to perform non-linear transformation to obtain the probabilities.

If we try to perform mean square error(MSE) then ,

MSE= (y- )2

MSE =

If we put this in our Gradient Descent optimizer then , we can see that , already the is non linear and if we try do square of that term then , it will become more non-linear or it is standing towards non-linear convex optimization problem. And, Gradient descent can’t work on this non convex optimization .Here it will be falling into that local minima Trap.

Hence, we can’t use MSE as a cost function for Logistic Regression. Instead of that we will use log loss function or binary cross entropy .