Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

Skill AP
Learn Anytime Anywhere
Andhra Pradesh State Skill Development Corporation

Machine Learning

Supervised Learning : Classification

CHAPTER -7

Supervised Learning : Classification

Introduction to Classification:

Classification may be defined as the process of predicting class or category from observed values or given data points. The categorized output can have the form such as “Black” or “White” or “spam” or “no spam”.

Mathematically, classification is the task of approximating a mapping function (f) from input variables (X) to output variables (Y). It basically belongs to the supervised machine learning in which targets are also provided along with the input data set.

An example of a classification problem can be the spam detection in emails. There can be only two categories of output, “spam” and “no spam”; hence this is a binary type classification.

To implement this classification, we first need to train the classifier. For this example, “spam” and “no spam” emails would be used as the training data. After successfully training the classifier, it can be used to detect an unknown email.

Types of Learners in Classification

We have two types of learners in respective to classification problems −

Lazy Learners

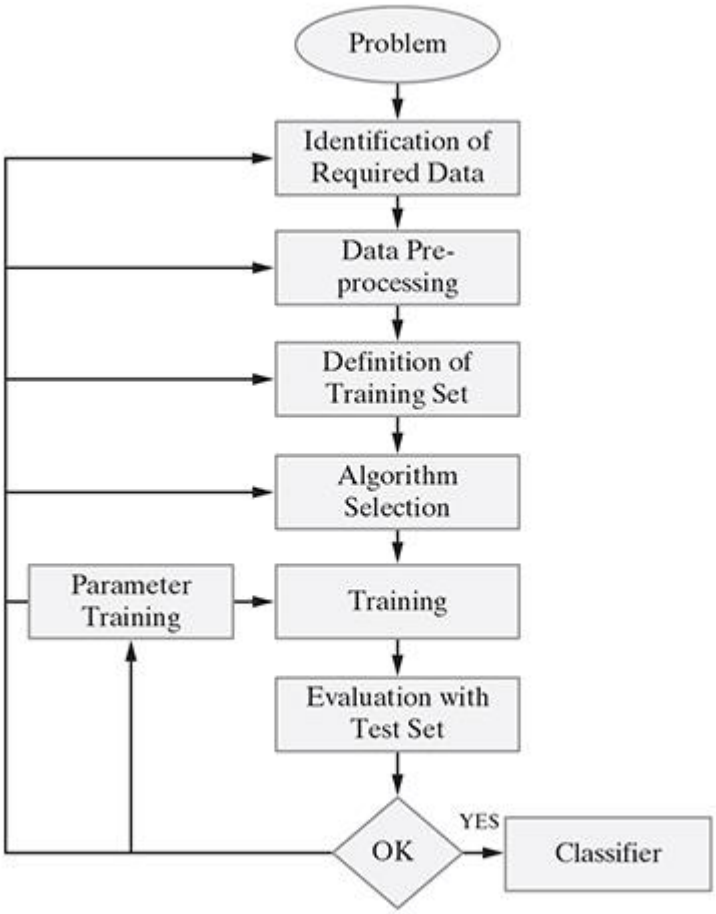
As the name suggests, such learners wait for the testing data to appear after storing the training data. Classification is done only after getting the testing data. They spend less time on training but more time on predicting. Examples of lazy learners are K-nearest neighbors and case-based reasoning.

Eager Learners

As opposed to lazy learners, eager learners construct classification models without waiting for the testing data to appear after storing the training data. They spend more time on training but less time on predicting. Examples of eager learners are Decision Trees, Naïve Bayes and Artificial Neural Networks (ANN).

Classification Steps:

First, there is a problem which is to be solved, and then, the required data (related to the problem, which is already stored in the system) is evaluated and pre-processed based on the algorithm. Algorithm selection is a critical point in supervised learning. The result after iterative training rounds is a classifier for the problem in hand



Problem Identification: Identifying the problem is the first step in the supervised learning model. The problem needs to be a well-formed problem,i.e. a problem with well-defined goals and benefit, which has a long-term impact.

Identification of Required Data: On the basis of the problem identified above, the required data set that precisely represents the identified problem needs to be identified/evaluated. For example: If the problem is to predict whether a tumour is malignant or benign, then the corresponding patient data sets related to malignant tumour and benign tumours are to be identified.

Data Pre-processing: This is related to the cleaning/transforming the data set. This step ensures that all the unnecessary/irrelevant data elements are removed. Data pre-processing refers to the transformations applied to the identified data before feeding the same into the algorithm. Because

the data is gathered from different sources, it is usually collected in a raw format and is not ready for immediate analysis. This step ensures that the data is ready to be fed into the machine learning algorithm.

Definition of Training Data Set: Before starting the analysis, the user should decide what kind of data set is to be used as a training set. In the case of signature analysis, for example, the training data set might be a single handwritten alphabet, an entire handwritten word (i.e. a group of the alphabets) or an entire line of handwriting (i.e. sentences or a group of words). Thus, a set of 'input meta-objects' and corresponding 'output meta-objects' are also gathered. The training set needs to be actively representative of the real-world use of the given scenario. Thus, a set of data input (X) and corresponding outputs (Y) is gathered either from human experts or experiments.

Algorithm Selection: This involves determining the structure of the learning function and the corresponding learning algorithm. This is the most critical step of supervised learning model. On the basis of various parameters, the best algorithm for a given problem is chosen.

Training: The learning algorithm identified in the previous step is run on the gathered training set for further fine tuning. Some supervised learning algorithms require the user to determine specific control parameters (which are given as inputs to the algorithm). These parameters (inputs given to algorithm) may also be adjusted by optimizing performance on a subset (called as validation set) of the training set.

Evaluation with the Test Data Set: Training data is run on the algorithm, and its performance is measured here. If a suitable result is not obtained, further training of parameters may be required.

The most common Classification Algorithms include :
1)Logistic Regression
2)Decision Trees
3)Ensemble Methods
4)Support Vector Machine (SVM)
5)Naive Bayes classifier
6)k-Nearest Neighbour (kNN)

Introduction to Logistic Regression :

Logistic regression is both classification and regression technique depending on the scenario used. Logistic regression (log it regression) is a type of regression analysis used for predicting the outcome of a categorical dependent variable similar to OLS regression. In logistic regression, dependent variable (Y) is binary (0,1) and independent variables (X) are continuous in nature. The probabilities describing the possible outcomes (probability that $Y = 1$) of a single trial are modelled as a logistic function of the predictor variables. In the logistic regression model, there is no $R2$ to gauge the fit of the overall model; however, a chi-square test is used to gauge how well the logistic regression model fits the data. The goal of logistic regression is to predict the likelihood that Y is equal to 1 (probability that $Y = 1$ rather than 0) given certain values of X. That is, if X and Y have a strong positive linear relationship, the probability that a person will have a score of $Y = 1$ will increase as values of X increase. So, we are predicting probabilities rather than the scores of the dependent variable.

For example, we might try to predict whether or not a small project will succeed or fail on the basis of the number of years of experience of the project manager handling the project. We presume that those project managers who have been managing projects for many years will be more likely to succeed. This means that as X (the number of years of experience of project manager) increases, the probability that Y will be equal to 1 (success of the new project) will tend to increase. If we take a hypothetical example in which 60 already executed projects were studied and the years of experience of project managers ranges from 0 to 20 years, we could represent this tendency to increase the probability that $Y = 1$ with a graph.

To illustrate this, it is convenient to segregate years of experience into categories (i.e. 0–8, 9–16, 17–24, 25–32, 33–40). If we compute the mean score on Y (averaging the 0s and 1s) for each category of years of experience, we will get something like

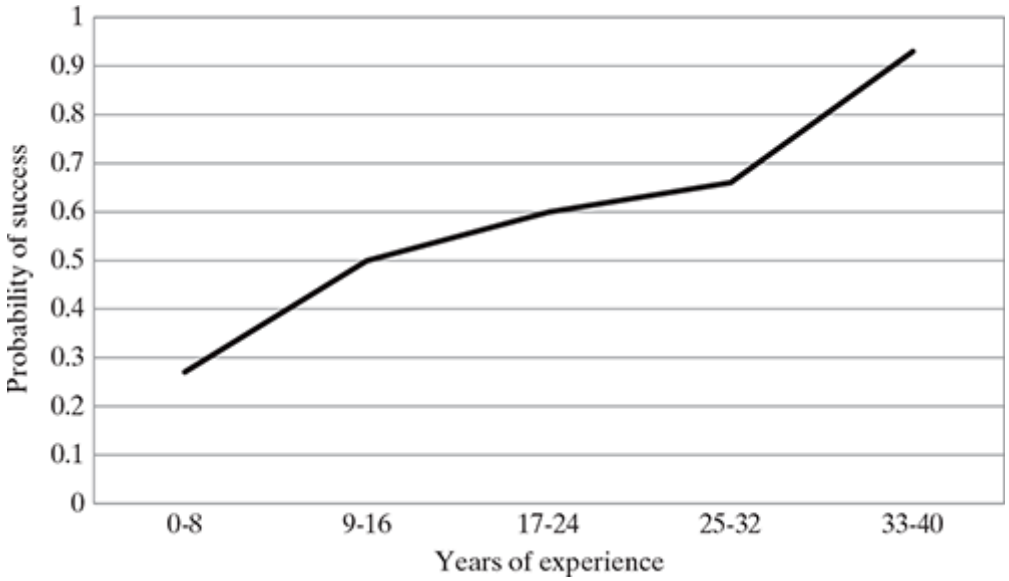
X	*Y*
0–8	0.27
9–16	0.5
17–24	0.6
25–32	0.66
33–40	0.93

When the graph is drawn for the above values of X and Y, it appears like the graph. As X increases, the probability that $Y = 1$ increases. In other words, when the project manager has more years of experience, a larger percentage of projects succeed. A perfect relationship represents a perfectly curved S rather than a straight line, as was the case in OLS regression. So, to model this relationship, we need some fancy algebra / mathematics that accounts for the bends in the curve.

An explanation of logistic regression begins with an explanation of the logistic function, which always takes values between zero and one. The logistic formulae are stated in terms of the probability that $Y = 1$, which is referred to as P. The probability that Y is 0 is $1 - P$.

$$\ln\left(\frac{P}{1-P}\right) = a + bX$$

$$\ln(p/1-p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \varepsilon$$



Logistic regression

The 'ln' symbol refers to a natural logarithm and $a + bX$ is the regression line equation. Probability (P) can also be computed from the regression equation. So, if we know the regression equation, we could, theoretically, calculate the expected probability that $Y = 1$ for a given value of X.

$$P = \frac{\exp(a + bX)}{1 + \exp(a + bx)} = \frac{e^{a+bx}}{1 + e^{a+bx}}$$

'exp' is the exponent function, which is sometimes also written as e.

Let us say we have a model that can predict whether a person is male or female on the basis of their height. Given a height of 150 cm, we need to predict whether the person is male or female.

We know that the coefficients of $a = -100$ and $b = 0.6$. Using the above equation, we can calculate the probability of male given a height of 150 cm or more formally

P(male|height = 150).

y = e^($a + b \times X$)/(1 + e^($a + b \times X$))
y = exp (−100 + 0.6 × 150)/(1 + EXP(−100 + 0.6 × X)
y = 0.000046
or a probability of near zero that the person is a male.

Assumptions in logistic regression

The following assumptions must hold when building a logistic regression model:

- There exists a linear relationship between logit function and independent variables
- The dependent variable *Y* must be categorical (1/0) and take binary value, e.g. if pass then *Y* = 1; else *Y* = 0
- The data meets the 'iid' criterion, i.e. the error terms, ε, are independent from one another and identically distributed
- The error term follows a binomial distribution [*n*, *p*]
    - *n* = # of records in the data
    - *p* = probability of success (pass, responder)

Loading a dataset:-

Data Collection

The quantity & quality of your data dictate how accurate our model is The outcome of this step is generally a representation of data (Guo simplifies to specifying a table) which we will use for training. Using pre-collected data, by way of datasets from Kaggle, UCI, etc.

Implementation of Data Preprocessing

Data preprocessing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for building and training Machine Learning models. In simple words, data preprocessing in Machine Learning is a data mining technique that transforms raw data into an understandable and readable format.

Steps in Data Preprocessing in Machine Learning :-

1. Acquire the dataset

To build and develop Machine Learning models, you must first acquire the relevant dataset. This dataset will be composed of data gathered from multiple and disparate sources which are then combined in a proper format to form a dataset. Dataset formats differ according to use cases. For instance, a business dataset will be entirely different from a medical dataset. While a business dataset will contain relevant industry and business data, a medical dataset will include healthcare-related data.

2. Import all the crucial libraries

Since Python is the most extensively used and also the most preferred library by Data Scientists around the world, we'll show you how to import Python libraries for data preprocessing in Machine Learning
The three core Python libraries used for this data preprocessing in Machine Learning are:

- NumPy – NumPy is the fundamental package for scientific calculation in Python. Hence, it is used for inserting any type of mathematical operation in the code. Using NumPy, you can also add large multidimensional arrays and matrices in your code.
- Pandas – Pandas is an excellent open-source Python library for data manipulation and analysis. It is extensively used for importing and managing the datasets. It packs in high-performance, easy-to-use data structures and data analysis tools for Python.
- Matplotlib – Matplotlib is a Python 2D plotting library that is used to plot any type of charts in Python. It can deliver publication-quality figures in numerous hard copy formats and interactive environments across platforms (IPython shells, Jupyter notebook, web application servers, etc.).

3. Import the dataset

In this step, you need to import the dataset/s that you have gathered for the ML project at hand. However, before you can import the dataset/s, you must set the current directory as the working directory.

4. Identifying and handling the missing values

In data preprocessing, it is pivotal to identify and correctly handle the missing values, failing to do this, you might draw inaccurate and faulty conclusions and inferences from the data. Needless to say, this will hamper your ML project.

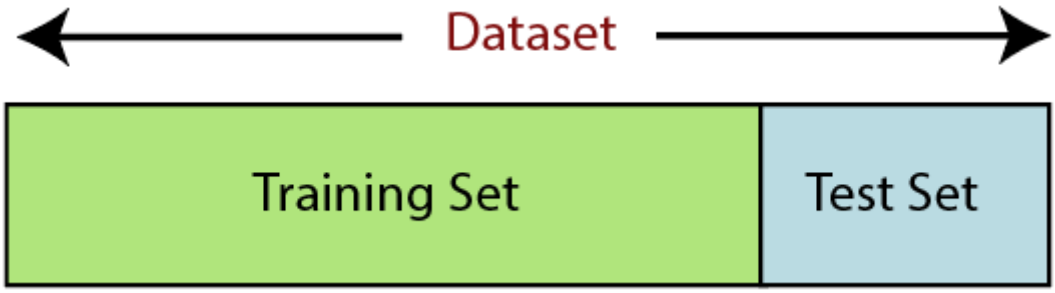
Basically, there are two ways to handle missing data:

- Deleting a particular row – In this method, you remove a specific row that has a null value for a feature or a particular column where more than 75% of the values are missing. However, this method is not 100% efficient, and it is recommended that you use it only when the dataset has adequate samples. You must ensure that after deleting the data, there remains no addition of bias.
- Calculating the mean – This method is useful for features having numeric data like age, salary, year, etc. Here, you can calculate the mean, median, or mode of a particular feature or column or row that contains a missing value and replace the result for the missing value. This method can add variance to the dataset, and any loss of data can be efficiently negated. Hence, it yields better results compared to the first method (omission of rows/columns). Another way of approximation is through the deviation of neighbouring values. However, this works best for linear data.

5. Encoding the categorical data

Categorical data refers to the information that has specific categories within the dataset. In the dataset cited above, there are two categorical variables – country and purchased.
Machine Learning models are primarily based on mathematical equations. Thus, you can intuitively understand that keeping the categorical data in the equation will cause certain issues since you would only need numbers in the equations.

Every dataset for Machine Learning models must be split into two separate sets – training set and test set.



7. Feature scaling

Feature scaling marks the end of the data preprocessing in Machine Learning. It is a method to standardize the independent variables of a dataset within a specific range. In other words, feature scaling limits the range of variables so that you can compare them on common grounds.

Evaluation - Modelling :

Methods for evaluating a model's performance are divided into 2 categories: namely, holdout and Cross-validation. Both methods use a test set (i.e data not seen by the model) to evaluate model performance. It's not recommended to use the data we used to build the model to evaluate it. This is because our model will simply remember the whole training set, and will therefore always predict the correct label for any point in the training set. This is known as overfitting.

Holdout

The purpose of holdout evaluation is to test a model on different data than it was trained on. This provides an unbiased estimate of learning performance.

In this method, the dataset is *randomly* divided into three subsets:

1. **Training set** is a subset of the dataset used to build predictive models.
2. **Validation set** is a subset of the dataset used to assess the performance of the model built in the training phase. It provides a test platform for fine-tuning a model's parameters and selecting the best performing model. Not all modeling algorithms need a validation set.
3. **Test set**, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits the training set much better than it fits the test set, overfitting is probably the cause.

The holdout approach is useful because of its speed, simplicity, and flexibility. However, this technique is often associated with high variability since differences in the training and test dataset can result in meaningful differences in the estimate of accuracy.

Cross-Validation

Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis**.**

The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds. The k is a user-specified number, usually with 5 or 10 as its preferred value. This is repeated k times, such that each time, one of the k subsets is used as the test set/validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of our model.

For instance, when performing five-fold cross-validation, the data is first partitioned into 5 parts of (approximately) equal size. A sequence of models is trained. The first model is trained using the first fold as the test set, and the remaining folds are used as the training set. This is repeated for each of these 5 splits of the data and the estimation of accuracy is averaged over all 5 trials to get the total effectiveness of our model.

As can be seen, every data point gets to be in a test set exactly once and gets to be in a training set k-1 times. This significantly reduces bias, as we're using most of the data for fitting, and it also significantly reduces variance, as most of the data is also being used in the test set. Interchanging the training and test sets also adds to the effectiveness of this method.

Model Evaluation Metrics

Model evaluation metrics are required to quantify model performance. The choice of evaluation metrics depends on a given machine learning task (such as classification, regression, ranking, clustering, topic modelling, among others). Some metrics, such as precision-recall, are useful for multiple tasks. Supervised learning tasks such as classification and regression constitutes a majority of machine learning applications. In this article, we focus on metrics for these two supervised learning models.

Classification Metrics

In this section we will review some of the metrics used in classification problems, namely:

- Classification Accuracy
- Confusion matrix
- Logarithmic Loss
- Area under curve (AUC)
- F-Measure

Classification Accuracy

Accuracy is a common evaluation metric for classification problems. It's the number of correct predictions made as a ratio of all predictions made. We use sklearn module to compute the accuracy of a classification task

Confusion Matrix

A confusion matrix provides a more detailed breakdown of correct and incorrect classifications for each class

Logarithmic Loss

Logarithmic loss (log loss) measures the performance of a classification model where the prediction input is a probability value between 0 and 1. Log loss increases as the predicted probability diverges from the actual label. The goal of machine learning models is to minimize this value. As such, smaller log loss is better, with a perfect model having a log loss of 0.

F1 Score

We now know that models can be classified as high precision or high recall models. Depending on the goal of the model. However, it is inconvenient to always have to carry two numbers around in order to make a decision about a model. Therefore it would be nice to combine recall and precision into a single score. This can be done by simply taking the average of precision and recall. This would be a bad idea as models that have a low precision or recall would still get a high score.

A better way of calculating a single score out of precision and recall is called the harmonic mean.

Harmonic Mean

It is a mathematical fact that the harmonic mean is always less than the arithmetic mean. The harmonic mean will produce a low score when either the precision or recall is very low.

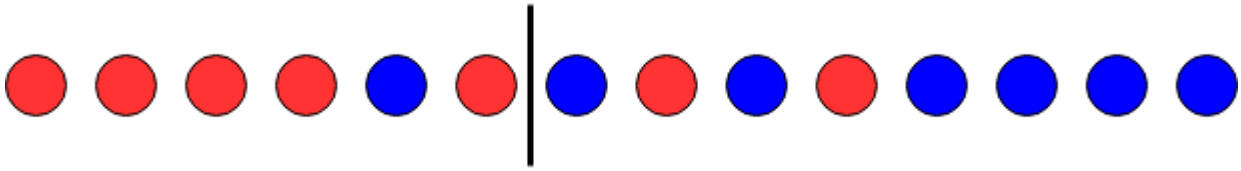
In Machine Learning model evaluation and validation, the harmonic mean is called the F1 Score.

F-1 Score = 2 * (Precision + Recall / Precision * Recall)

Roc Curve(Receiver Operating Characteristic)

Splitting the data N times and plotting the values. For each split, we calculate True positive and true negative.
The closer the value under the curve to 1 the better the model is.

Consider a one-dimensional dataset consisting of the following 14 points.
In order to plot a ROC curve, we would need to split the data N times and calculate the True Positive Rate and False Positive Rate for each split.

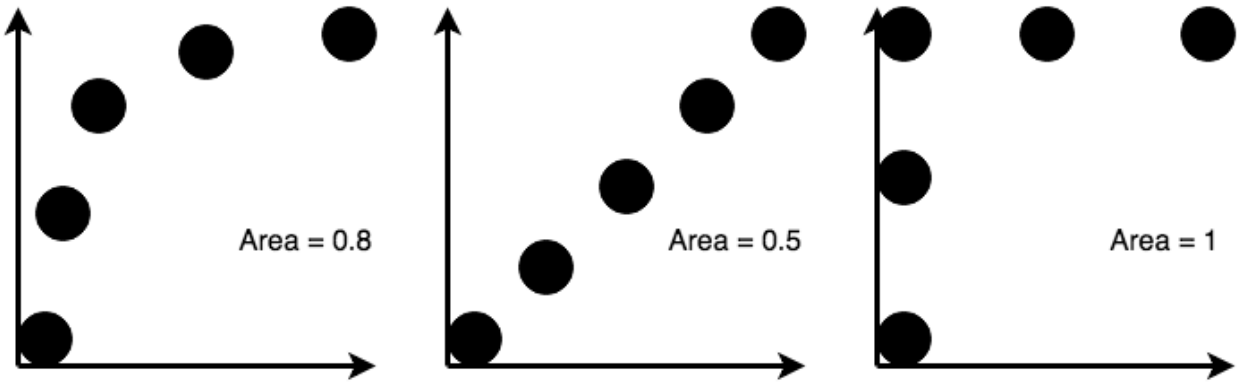


True Positive Rate = True Positives / All Positives = 6/7 = 0,857

False Positive Rate = False Positives / All Negatives = 2/7 = 0,286

The N-size set of (True Positive Rate, False Positive Rate) can then be plotted.

- A random model will have a score of around 0.5
- A good model will have a score closer to 1
- A perfect model will have a score of 1



Precision and Recall

As we can see models can be fundamentally different depending on what they are solving. Therefore models can have totally different priorities. In order to measure these differences in priorities, we have two metrics that can be used. The metrics are called Precision and Recall.

Precision

The precision metric can be calculated as follows.
True positive / True positive + False Negative
In other words out of e.g. the patients that the model classified as sick, how many did the model correctly classify as sick?

Recall

The recall metric can be calculated as follows
True positive / True positive + False positive
The recall metric is kind of the opposite of Precision. Therefore for the model classifying patients as sick or not sick this would answer the question. Out of all sick patients, how many did the model correctly classify as sick?

Introduction to Softmax Regression

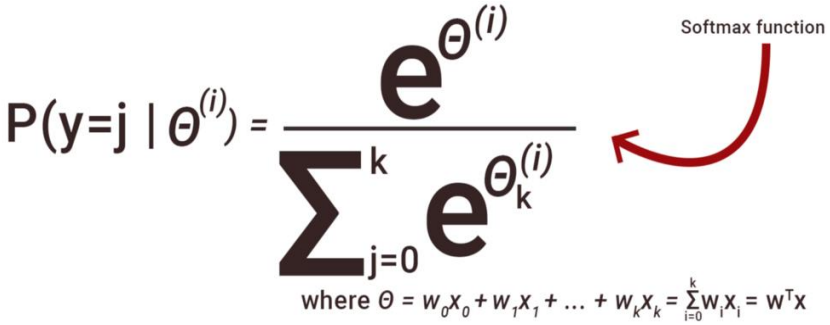
The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model.

This is why softmax is sometimes referred to as a multinomial logistic regression
The function is usually used to compute losses that can be expected when training a data set. Known use-cases of Softmax regression are in **discriminative models** such as **Cross-Entropy** and **Noise Contrastive Estimation**. These are only two among various techniques that attempt to optimize the current training set to increase the likelihood of predicting the correct word or sentence. (We will touch upon the aforementioned techniques in the next few posts, so stay tuned for those.)

If you look at it from the outset, the definition may sound off as trivial but in the realm of Machine Learning and NLP, this regression function has been useful as a *baseline* comparator. Researchers who design new solutions have to carry out experimentation keeping the Softmax results as a reference.

Notation

The classifier function involves some high-level notation which we are going to dive into next. The picture below illustrates how a Softmax function looks like. Let's try to understand it piece by piece.



1.Given a net input parameter in the form of a **one-hot encoded** matrix **θ,** our objective is to predict if the trained set of features ***x;*** *each* with its own set of weights, are a class of ***j***. A one-hot matrix consists of binary values with the number 1 representing an element in the i^{th} position of the column while the rest are 0s (a relatively large matrix runs the risk of scarcity, which we will talk about in the next post).

2.In the formula we compute the **exponential** of the input parameter and the **sum of exponential parameters** of all existing values in the inputs. Our output for the Softmax function is the ratio of the exponential of the parameter and the sum of exponential parameters.

3.**θ**, on a high level is the sum of the score of each occurring element in the vector. In a generalized form we say that θ is the transpose of the weights matrix ***w***, multiplied by the feature matrix ***x.***

4.The term w_0x_0 is the bias that needs to be added on every iteration.

Implementation

Implementing the code is extraordinarily easy and the fun part is that it's only one line considering we have the necessary Python helper functions at our disposal. In the problem implementation below I have used both **Numpy**, **Pandas** libraries to write the Softmax function as explained in the previous section.

Evaluation

After training the Softmax regression model, given any example features, we can predict the probability of each output class. Normally, we use the class with the highest predicted probability as the output class. The prediction is correct if it is consistent with the actual class (label). In the next part of the experiment, we will use accuracy to evaluate the model's performance. This is equal to the ratio between the number of correct predictions and the total number of predictions.