<https://www.javatpoint.com/regression-vs-classification-in-machine-learning#:~:text=Difference%20between%20Regression%20and%20Classification%20%20%20,the%20de%20...%20%202%20more%20rows%20>

Regression vs. Classification

Regression and Classification algorithms are Supervised Learning algorithms. Both the algorithms are used for prediction in Machine learning and work with the labeled datasets. But the difference between both is how they are used for different machine learning problems.

The main difference between Regression and Classification algorithms that Regression algorithms are used to **predict the continuous** values such as price, salary, age, etc. and Classification algorithms are used to **predict/Classify the discrete values** such as Male or Female, True or False, Spam or Not Spam, etc.

## **Classification:**

Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In Classification, a computer program is trained on the training dataset and based on that training, it categorizes the data into different classes.

The task of the classification algorithm is to find the mapping function to map the input(x) to the discrete output(y).

**Example:** The best example to understand the Classification problem is Email Spam Detection. The model is trained on the basis of millions of emails on different parameters, and whenever it receives a new email, it identifies whether the email is spam or not. If the email is spam, then it is moved to the Spam folder.

**Types of ML Classification Algorithms:**

Classification Algorithms can be further divided into the following types:

* Logistic Regression
* K-Nearest Neighbours
* Support Vector Machines
* Kernel SVM
* Naïve Bayes
* Decision Tree Classification
* Random Forest Classification

**Regression:**

Regression is a process of finding the correlations between dependent and independent variables. It helps in predicting the continuous variables such as prediction of **Market Trends**, prediction of House prices, etc.

The task of the Regression algorithm is to find the mapping function to map the input variable(x) to the continuous output variable(y).

**Example:** Suppose we want to do weather forecasting, so for this, we will use the Regression algorithm. In weather prediction, the model is trained on the past data, and once the training is completed, it can easily predict the weather for future days.

**Types of Regression Algorithm:**

* Simple Linear Regression
* Multiple Linear Regression
* Polynomial Regression
* Support Vector Regression
* Decision Tree Regression
* Random Forest Regression

<https://www.analyticsvidhya.com/blog/2020/08/bias-and-variance-tradeoff-machine-learning/>

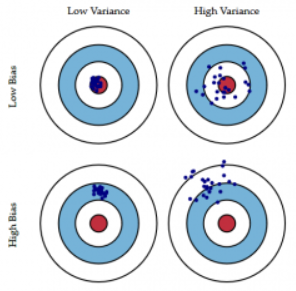
## What is Bias?

Bias is the difference between the Predicted Value and the Expected Value. When there is a high bias error, it results in a very simplistic model that does not consider the variations very well. Since it does not learn the training data very well, it is called **Underfitting.**

## What is a Variance?

Contrary to bias, the Variance is when the model takes into account the fluctuations in the data i.e. the noise as well. Since in the case of high variance, the model learns too much from the training data, it is called **overfitting.**

The balance between the Bias error and the Variance error is the **Bias-Variance Tradeoff**. The following bulls-eye diagram explains the tradeoff better:



## Conclusion

To summarize, in this article, we learned that an ideal model would be one where both the bias error and the variance error are low. However, we should always aim for a model where the model score for the training data is as close as possible to the model score for the testing data.

That’s where we figured out how to choose a model that is not too complex (High variance and low bias) which would lead to overfitting and nor too simple(High Bias and low variance) which would lead to underfitting.

Bias and Variance plays an important role in deciding which predictive model to use. I hope this article explained the concept well.

<https://arifromadhan19.medium.com/part-1-regression-and-classification-model-evaluation-bc7f6ab3b4dd>

Part 1: Regression and Classification Model Evaluation

Regression

* R-Squared – It is also known as the **coefficient of determination**. This metric gives an indication of how good a model fits a given dataset. It indicates how close the [regression line](https://www.studytonight.com/post/classification-problem-introduction-to-logistic-regression) (i.e the predicted values plotted) is to the actual data values. The **R squared value lies between 0 and 1** where 0 indicates that this model doesn't fit the given data and 1 indicates that the model fits perfectly to the dataset provided.

import numpy as np

X = np.random.randn(100)

y = np.random.randn(60) # y has nothing to do with X whatsoever

from sklearn.linear\_model import LinearRegression

from sklearn.cross\_validation import cross\_val\_score

scores = cross\_val\_score(LinearRegression(), X, y,scoring='r2')

* RMSE : Root Mean Squared Error [0- infinity] - RMSE is the standard deviation of the errors which occur when a prediction is made on a dataset. This is the same as MSE (Mean Squared Error) but the root of the value is considered while determining the accuracy of the model.

from sklearn.metrics import mean\_squared\_error

from math import sqrt

actual\_values = [3, -0.5, 2, 7]

predicted\_values = [2.5, 0.0, 2, 8]

mean\_squared\_error(actual\_values, predicted\_values)

# taking root of mean squared error

root\_mean\_squared\_error = sqrt(mean\_squared\_error)

* Mean Squared Error or MSE - MSE is calculated by taking the average of the square of the difference between the original and predicted values of the data.

from sklearn.metrics import mean\_squared\_error

actual\_values = [3, -0.5, 2, 7]

predicted\_values = [2.5, 0.0, 2, 8]

mean\_squared\_error(actual\_values, predicted\_values)

* MAE : Mean Absolute Error [0- infinity] – We know that an error basically is the absolute difference between the actual or true values and the values that are predicted. Absolute difference means that if the result has a negative sign, it is ignored. Hence, **MAE = True values – Predicted values**

from sklearn.metrics import mean\_absolute\_error

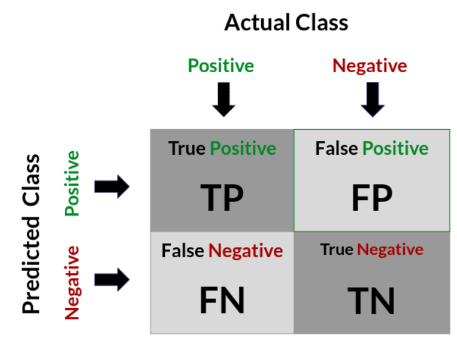
# predicting home prices in some area

predicted\_home\_prices = mycity\_model.predict(X)

mean\_absolute\_error(y, predicted\_home\_prices)

Classification

Before discussing model evaluation on classification, we should understand the confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.



Positive (P): Observation is positive (for example: is an apple).

Negative (N): Observation is not positive (for example: is not an apple).

True Positive (TP): Observation is positive, and is predicted to be positive.

False Negative (FN): Observation is positive, but is predicted negative.

True Negative (TN): Observation is negative, and is predicted to be negative.

False Positive (FP): Observation is negative, but is predicted positive.

* Accuracy

Accuracy = (TP + TN) / (TP + TN + FP + FN)  
Accuracy = (True positive+ True negative) /Total count of elements

* Precision

Precision = (TP) / (TP + FP)  
Precision = (True positive) / (True positive + False positive)

* Recall

Recall = (TP) / (TP + FN)  
Recall = (True positive) / (True positive + False negative)

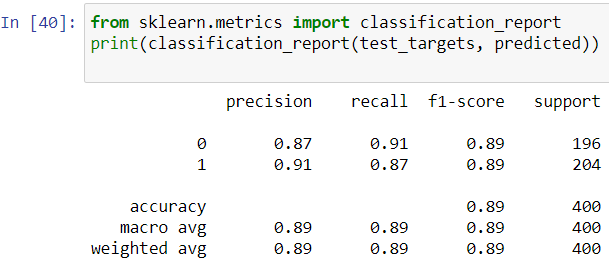
* Specificity

Specificity = (TN) / (TN + FP)  
Specificity = (True negative) / (True negative + False positive)

* F1 Score

F1 Score measure provides a way to combine both precision and recall into a single measure that captures both properties.

F1 Score = 2 \* (Recall\*Precision) / (Recall + Precision)



Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures (series / dataframes)

<https://www.tutorialspoint.com/python_pandas/python_pandas_series.htm>

<https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm>

# Python Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

<https://www.geeksforgeeks.org/python-numpy/>

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays.

<https://www.tutorialspoint.com/matplotlib/index.htm>

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of **[matplotlib](https://www.geeksforgeeks.org/python-introduction-matplotlib/" \t "_blank)** library and also closely integrated to the data structures from [pandas](https://www.geeksforgeeks.org/introduction-to-pandas-in-python/).

<https://www.geeksforgeeks.org/python-seaborn-lmplot-method/>

Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark community released a tool, PySpark. Using PySpark, you can work with RDDs in Python programming language also. It is because of a library called Py4j that they are able to achieve this. This is an introductory tutorial, which covers the basics of Data-Driven Documents and explains how to deal with its various components and sub-components.

<https://www.tutorialspoint.com/pyspark/index.htm>

Neo4j is the world's leading open source Graph Database which is developed using Java technology. It is highly scalable and schema free (NoSQL). A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. It is composed of two elements - nodes (vertices) and relationships (edges). Graph database is a database used to **model the data in the form of graph. In here, the nodes of a graph depict the entities while the relationships depict the association of these nodes**.

Neo4j is a popular Graph Database. Other Graph Databases are Oracle NoSQL Database, OrientDB, HypherGraphDB, GraphBase, InfiniteGraph, and AllegroGraph.

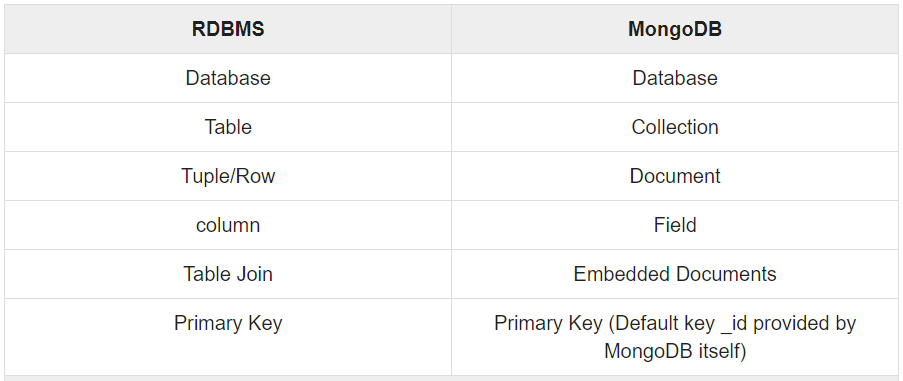
<https://www.tutorialspoint.com/neo4j/neo4j_overview.htm>

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is **based on Hadoop MapReduce** and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. **The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.**

Spark is designed to cover a wide range of workloads such as **batch applications**, iterative algorithms, interactive queries and **streaming**. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

<https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm>

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.



## Advantages of MongoDB over RDBMS

* **Schema less** − MongoDB is a document database in which one collection holds different documents. **Number of fields, content and size of the document can differ from one document to another**.
* Structure of a single object is clear.
* No complex joins.
* Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.

<https://www.tutorialspoint.com/mongodb/mongodb_overview.htm>

**Machine Learning Algorithms:**

* Scikit-learn,
* Keras / Tensorflow,
* Fbprophet,
* Lightgbm,
* Xgboost,
* Shap / Lime

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

**Features**

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

**Supervised Learning algorithms** − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

**Unsupervised Learning algorithms** − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

**Clustering** − This model is used for grouping unlabeled data.

**Cross Validation** − It is used to check the accuracy of supervised models on unseen data.

**Dimensionality Reduction** − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

**Ensemble methods** − As name suggest, it is used for combining the predictions of multiple supervised models.

**Feature extraction** − It is used to extract the features from data to define the attributes in image and text data.

**Feature selection** − It is used to identify useful attributes to create supervised models.

**Open Source** − It is open source library and also commercially usable under BSD license.

<https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm>

Keras is compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. => useful in image recognition, language modelling, and occasionally sequence prediction.

<https://www.tutorialspoint.com/tensorflow/tensorflow_keras.htm>

Prophet is an open-source tool from Facebook used for forecasting time series data which helps businesses understand and possibly predict the market. It is based on a decomposable additive model where non-linear trends are fit with seasonality, it also takes into account the effects of holidays. Before we head right into coding, let’s learn certain terms that are required to understand this.

**Trend:** shows the tendency of the data to increase or decrease over a long period of time and it filters out the seasonal variations.

**Seasonality:** variations that occur over a short period of time and is not prominent enough to be called a “trend”.

<https://www.geeksforgeeks.org/time-series-analysis-using-facebook-prophet/>

Explainable AI (Lime, Shap) has made the black-box model to be of High Accuracy and High Interpretable in nature for business use cases across industries and making decisions for business stakeholders to understand better.

Lime (Local Interpretable Model-agnostic Explanations) helps to illuminate a machine learning model and to make its predictions individually comprehensible. The method explains the classifier for a specific single instance and is therefore suitable for local consideration.

SHAP stands for SHapley Additive exPlanations. The core idea behind Shapley value-based explanations of machine learning models is to use fair allocation results from cooperative game theory to allocate credit for a model’s output f(x)f(x) among its input features. In order to connect game theory with machine learning models it is necessary to both match a model’s input features with players in a game, and also match the model function with the rules of the game.

Importance of Explainable AI

* Model Behavior
* Transparency
* making better decisions
* can explain any model be it black-box model or Deep learning
* it’s bridge gap and help to use more robust model(for better accuracy and explainability)
* Trustiness
* Model Debugging

SHAP and LIME are both popular Python libraries for model explainability. SHAP leverages the idea of model feature influence scoring. The technical definition of a Shapley value is the “average marginal contribution of a feature value over all possible coalitions.”.

<https://www.analyticsvidhya.com/blog/2020/10/unveiling-the-black-box-model-using-explainable-ai-lime-shap-industry-use-case/>

**Bagging and Boosting | Data Mining**

Bagging and Boosting are two types of Ensemble Learning. These two decrease the variance of single estimate as they combine several estimates from different models. So the result may be a model with higher stability.

* If the difficulty of the single model is over-fitting, then Bagging is the best option.
* If the problem is that the single model gets a very low performance, Boosting could generate a combined model with lower errors as it optimises the advantages and reduces pitfalls of the single model.

Similarities Between Bagging and Boosting –

1. Both are ensemble methods to get N learners from 1 learner.
2. Both generate several training data sets by random sampling.
3. Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
4. Both are good at reducing variance and provide higher stability.

**Differences Between Bagging and Boosting**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **BAGGING** | **BOOSTING** |
| 1. | Simplest way of combining predictions that belong to the same type. | A way of combining predictions that belong to the different types. |
| 2. | Aim to decrease variance, not bias. | Aim to decrease bias, not variance. |
| 3. | Each model receives equal weight. | Models are weighted according to their performance. |
| 4. | Each model is built independently. | New models are influenced by performance of previously built models. |
| 5. | Different training data subsets are randomly drawn with replacement from the entire training dataset. | Every new subsets contains the elements that were misclassified by previous models. |
| 6. | Bagging tries to solve over-fitting problem. | Boosting tries to reduce bias. |
| 7. | If the classifier is unstable (high variance), then apply bagging. | If the classifier is stable and simple (high bias) the apply boosting. |
| 8. | Random forest. | Gradient boosting. |

<https://www.geeksforgeeks.org/comparison-b-w-bagging-and-boosting-data-mining/>

<https://towardsdatascience.com/xgboost-lightgbm-and-other-kaggle-competition-favorites-6212e8b0e835>

*Xgboost and lightGBM are both subtypes/specific instances of the GBDT (****Gradient boosted decision tree****) algorithm. Though they both implement the same underlying algorithm, they each introduce various tricks to make training more efficient or to improve performance.*

**XGBoost, LightGBM, and Other Kaggle Competition Favorites**

Kaggle is the data scientist’s go-to place for datasets, discussions, and perhaps most famously, competitions with prizes of tens of thousands of dollars to build the best model.

With all the flurried research and hype around deep learning, one would expect **neural network** solutions to dominate the leaderboards. It turns out, however, that neural networks — while indeed very powerful algorithms — have very limited applications, being useful really only in image recognition, language modelling, and occasionally sequence prediction.

Instead, top winners of Kaggle competitions routinely use gradient boosting. It’s worth looking at the intuition of this fascinating algorithm and why it has become so popular among Kaggle winners.

Decision trees are relatively weak on their own — predictions are formed based solely on yes/no questions. The feature space is split into hypercubes, which may not be appropriate for many datasets that cannot be separated by vertical and horizontal planes. However, ensembles of trees that combine learned insights from different models can be very powerful.

Traditional ensemble models — the Random Forest algorithm — form ensembles by training several trees on random subsets of the data. This allows different trees to ‘specialize’ on certain parts of the data while considering the inputs of other trees when making a prediction on an input.

Random Forest can be effective, but it is expensive to train because trees are added completely at random. There is no clear difference between training twenty trees and fifteen trees, and the objective of each tree is to optimize its own performance on its data subset, not necessarily the performance of the ensemble as a whole.

Gradient Boosting is another tree-based ensemble method that centers around the intelligent construction of ensembles. Each tree’s purpose is not to maximize its own performance on a randomly delegated subset of data, but to pick up on where the previous tree lacks.

Gradient boosting can be used for classification and regression, but it also works well with ranking. Yahoo and Yandex — two commercial search engines — use variants of gradient boosting to produce fast search results.

XGBoost, or Extreme Gradient Boosting, is one of (perhaps, the) most popular gradient boosting implementation libraries, with support in several languages, including Python, R, Julia, C++, Scala, Perl, and Java.

LightGBM (Light Gradient Boosting Machine) is another top contender in gradient boosting implementations. Produced by Microsoft, its first stable version was released in 2017, three years after the release of XGBoost.

It boasts many of XGBoost’s advantages, including sparse optimization, parallel training, multiple loss functions, regularization, bagging, and early stopping. By most accounts LightGBM is faster than XGBoost, and LightGBM’s creators claim it is twenty times faster with the same performance. One important difference between the two lies in the construction of trees.

Whereas most implementations grow a tree level-wise — row by row — LightGBM grows trees leaf-wise, in that it chooses the leaf it believes will yield the largest decrease in loss. This method can yield more complex and intricate structures, which usually leads to a lower loss but may overfit.

The competitions in which gradient boosting is applied are always tabular data: for example, forecasting time series or classifying a game as a win or a loss, because these phenomena are often, at their root, tree-based. With some additional embellishment — boosting, bagging, regularization, sparsity optimization, parallel computing, shrinkage — emerges a powerful, but not too powerful model that models hierarchical data in a hierarchical fashion.