Classify Cow Activity Prediction Documentation

Aim:

Classify the Cow’s activities into 9 categories based on Data Collected from IMU Sensors.

Objective:

Predicting the Cow Position Activity to help and focus on each cow by its there plenty of positions, like drinking position, standing, eating, lying, ruminating\_lying, ruminating\_standing and so on.

Data Source:

IMU (Inertial Measurement Unit) Data (Accelerometer, Gyroscope, Magnetometer).

What is IMU ?

The IMU data’s are Accelerometer, Gyroscope, Magnetometer in the form of Inertial measurement unit, used to describe a collection of measurement tools, when installed in some device, catches movement with the help of accelerometer, gyroscope and magnetometer, in 3d space.

Variables Names:

* + acc\_x,acc\_y,acc\_z: accelerometer output for all 3 dimensions movement.
  + gyr\_x,gyr\_y,gyr\_z: gyroscope outputs, it measures rotation, rotation rate (angular velocity).
  + mag\_x,mag\_y,mag\_z: magnetometer outputs, catches magnetic field around the device.
  + All three (Acc, Gyr, Mag) gives output in different SI Units i.e The scale for all three are different, so Data must be normalized.

Classes and their Encoded values:

eating = 1

drinking = 2

walking = 3

standing =4

lying = 5

ruminating standing = 6

ruminating lying = 7

grooming = 8

idle/other = 9

The Initial Processing:

* Uploading the .csv datafile in our own path and executed regarding the dataset.
* We can renaming and reading the file.
* To get initiative level import all the built-in common libraries to the regarding the project.
* To determine numpy as np, pandas as pd, matploat lib as pyplt as plt, %matplotlib inline and seaborn as sns those effective libraries functions prompted output we can get it.
* Initially we should stimulate the Data information through out this filename.head(), isnull().sum(), filename.describe() and shaping the data set.

Visualizing the Package:

To demonstrates the Top 10 Python Libraries for Data Visualization that are commonly used these days.

* Matplotlib. ...
* Plotly. ...
* Seaborn. ...
* GGplot. ...
* Altair. ...
* Bokeh. ...
* Pygal. ...
* Geoplotlib.

Data Visualization is an extremely important part of Data Analysis. After all, there is no better way to understand the hidden patterns and layers in the data than seeing them in a visual format! Don’t trust me? Well, assume that you analyzed your company data and found out that a particular product was consistently losing money for the company. Your boss may not pay that much attention to a written report but if you present a line chart with the profits as a red line that is consistently going down, then your boss may pay much more attention! This shows the power of **Data Visualization**!

Data visualization charts like **bar charts, scatterplots, line charts, geographical maps,** etc. are extremely important. They tell you information just by looking at them whereas normally you would have to read spreadsheets or text reports to understand the data. And [**Python**](https://www.geeksforgeeks.org/python-programming-language/) is one of the most popular programming languages for data analytics as well as data visualization. There are several libraries available in recent years that create beautiful and complex data visualizations. These libraries are so popular because they allow analysts and statisticians to create visual data models easily according to their specifications by conveniently providing an interface, data visualization tools all in one place! This article demonstrates the **Top 10 Python Libraries for Data Visualization** that are commonly used these days.

Matplotlib:

It is the most popular and widely-used plotting library in the Python community. It comes with an interactive environment across multiple platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, etc. It can be used to embed plots into applications using various  GUI toolkits like Tkinter, GTK+, wxPython, Qt, etc. So you can use Matplotlib to create plots, bar charts, pie charts, histograms, scatterplots, error charts, power spectra, stemplots, and whatever other visualization charts you want! The Pyplot module also provides a MATLAB-like interface that is just as versatile and useful as MATLAB while being free and open source.

Plotly:

It is a free open-source graphing library that can be used to form data visualizations. Plotly (plotly.py) is built on top of the Plotly JavaScript library (plotly.js) and can be used to create web-based data visualizations that can be displayed in Jupyter notebooks or web applications using Dash or saved as individual [HTML](https://www.geeksforgeeks.org/html-tutorials/) files. Plotly provides more than 40 unique chart types like scatter plots, histograms, line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. Plotly also provides contour plots, which are not that common in other data visualization libraries. In addition to all this, Plotly can be used offline with no internet connection.

Seaborn:

Seaborn is a Python data visualization library that is based on Matplotlib and closely integrated with the NumPy and pandas data structures. Seaborn has various dataset-oriented plotting functions that operate on data frames and arrays that have whole datasets within them. Then it internally performs the necessary statistical aggregation and mapping functions to create informative plots that the user desires. It is a high-level interface for creating beautiful and informative statistical graphics that are integral to exploring and understanding data. The Seaborn data graphics can include bar charts, pie charts, histograms, scatterplots, error charts, etc. Seaborn also has various tools for choosing color palettes that can reveal patterns in the data.

GGplot:

Ggplot is a Python data visualization library that is based on the implementation of ggplot2 which is created for the programming language R. Ggplot can create data visualizations such as bar charts, pie charts, histograms, scatterplots, error charts, etc. using high-level API. It also allows you to add different types of data visualization components or layers in a single visualization. Once ggplot has been told which variables to map to which aesthetics in the plot, it does the rest of the work so that the user can focus on interpreting the visualizations and take less time in creating them. But this also means that it is not possible to create highly customized graphics in ggplot. Ggplot is also deeply connected with pandas so it is best to keep the data in DataFrames.

Altair:

Altair is a statistical data visualization library in Python. It is based on Vega and Vega-Lite which are a sort of declarative language for creating, saving, and sharing data visualization designs that are also interactive. Altair can be used to create beautiful data visualizations of plots such as bar charts, pie charts, histograms, scatterplots, error charts, power spectra, stemplots, etc. using a minimal amount of coding. Altair has dependencies which include python 3.6, entrypoints, jsonschema, NumPy, Pandas, and Toolz which are automatically installed with the Altair installation commands. You can open Jupyter Notebook or JupyterLab and execute any of the code to obtain that data visualizations in Altair. Currently, the source for Altair is available on GitHub.

Bokeh:

Bokeh is a data visualization library that provides detailed graphics with a high level of interactivity across various datasets, whether they are large or small. Bokeh is based on The Grammar of Graphics like ggplot but it is native to Python while ggplot is based on ggplot2 from R. Data visualization experts can create various interactive plots for modern web browsers using bokeh which can be used in interactive web applications, HTML documents, or JSON objects. Bokeh has 3 levels that can be used for creating visualizations. The first level focuses only on creating the data plots quickly, the second level controls the basic building blocks of the plot while the third level provides full autonomy for creating the charts with no pre-set defaults. This level is suited to the data analysts and IT professionals that are well versed in the technical side of creating data visualizations.

PygaL:

Pygal is a Python data visualization library that is made for creating sexy charts! (According to their website!) While Pygal is similar to Plotly or Bokeh in that it creates data visualization charts that can be embedded into web pages and accessed using a web browser, a primary difference is that it can output charts in the form of SVG’s or Scalable Vector Graphics. These SVG’s ensure that you can observe your charts clearly without losing any of the quality even if you scale them. However, SVG’s are only useful with smaller datasets as too many data points are difficult to render and the charts can become sluggish.

Geoplotlib:

Most of the data visualization libraries don’t provide much support for creating maps or using geographical data and that is why geoplotlib is such an important Python library. It supports the creation of geographical maps in particular with many different types of maps available such as dot-density maps, choropleths, symbol maps, etc. One thing to keep in mind is that requires NumPy and pyglet as prerequisites before installation but that is not a big disadvantage. Especially since you want to create geographical maps and geoplotlib is the only excellent option for maps out there!

Importing The Classification Packages:

Now we have some of the packages are:

* Linear Regression (Supervised Learning).

Single Linear Regression. (To fit a straight Line).

Multiple Linear Regression. (some other like parabola)

* Logistic Regression (Supervised Learning)
* (Sigmoid function implementation).
* Binary Classification.
* Multi class classification.
* Multinominal Classification.
* Decision Tree

Ensemble method.

Entropy method.

Gini Index recovery method.

* Random Forest (Supervised Learning)

Top down approach.

* SVM (Support Vector Machine) (Supervised Learning)

For centroid straight line to support two vertical line.

The Nine types of Neck Rotation:

The nine csv files were first loaded onto google drive , the drive was mounted onto google colab notebook , each file was renamed and read. There is a plenty of dataset that is one crore more and above.

Classes and their Encoded values:

* eating = 1
* drinking = 2
* walking = 30
* standing =4
* lying = 5
* ruminating standing = 6
* ruminating lying = 7
* grooming = 8
* idle/other =9

All the files were concatenated into a single dataframe called df. This dataframe had 12263524 records and 11 features.

Since the dataframe df was very large in number , 2 smaller dataframes df1 and df2 of 10000 records each were created using random sampling. Checked if the count of records under each feature of the 2 smaller databases df1 and df2 matched with that of the original dataframe df using pie chart. Since it was closely matching, it was decided to undertake further workings using df1 and df2 is containing 10000 records.

Import the model Parameters:

* Confusion Matrix
* f1-Score
* Precision score
* Recall score
* Accuracy score

Confusion Matrix:

Confusion matrix is one of the easiest and most intuitive metrics used for finding the accuracy of a classification model, where the output can be of two or more categories. This is the most popular method used to evaluate [logistic regression](https://intellipaat.com/blog/what-is-logistic-regression/).

[What is a Confusion Matrix?](https://intellipaat.com/blog/confusion-matrix-python/#What-Is-a-Confusion-Matrix)

[True Positive](https://intellipaat.com/blog/confusion-matrix-python/#True-Positive)

[False Negative](https://intellipaat.com/blog/confusion-matrix-python/#False-Negative)

[False Positive](https://intellipaat.com/blog/confusion-matrix-python/#False-Positive)

[True Negative](https://intellipaat.com/blog/confusion-matrix-python/#True-Negative)

[Understanding various performance metrics](https://intellipaat.com/blog/confusion-matrix-python/#Understanding-various-performance-metrics):

[Accuracy or Classification Accuracy](https://intellipaat.com/blog/confusion-matrix-python/#Accuracy)

[Precision](https://intellipaat.com/blog/confusion-matrix-python/#Precision)

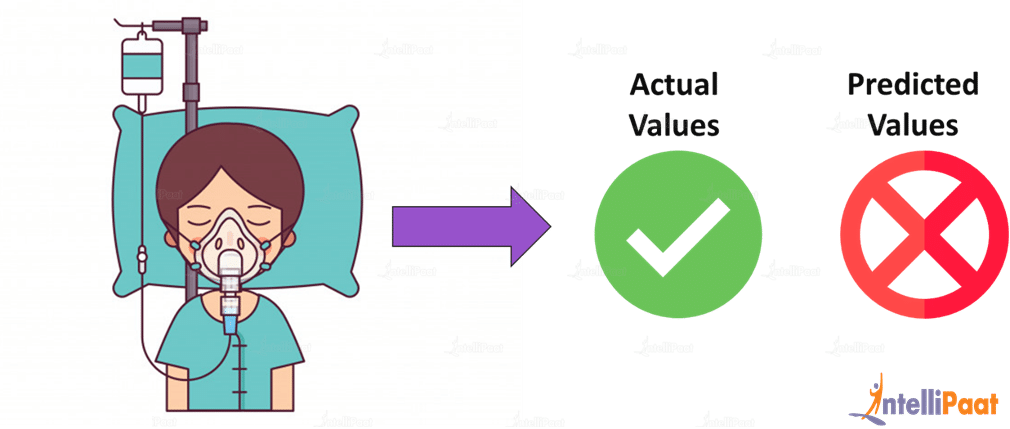
[Recall or Sensitivity](https://intellipaat.com/blog/confusion-matrix-python/#Recall-or-Sensitivity)

[Specificity](https://intellipaat.com/blog/confusion-matrix-python/#Specificity)

[F1 Score](https://intellipaat.com/blog/confusion-matrix-python/#F1-Score)

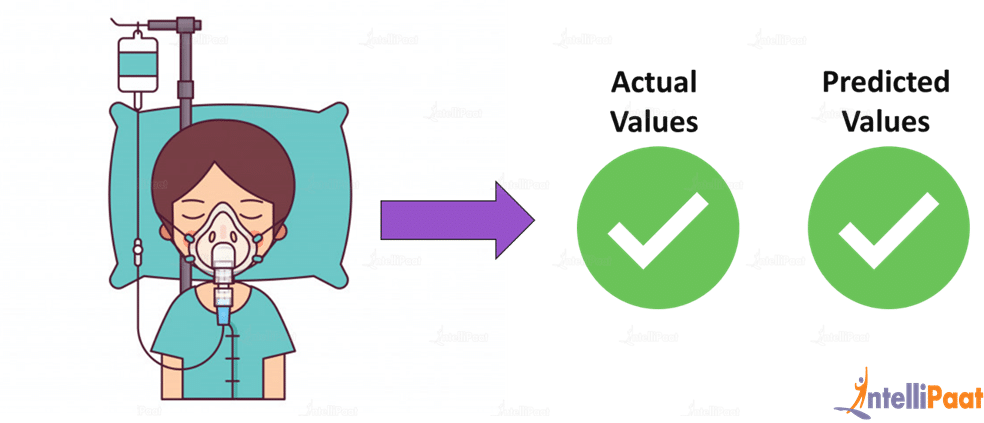
### ****False Negative****

In false negative, the actual value is true, but the predicted value is false, which means that the patient has cancer, but the model predicted that the patient did not have cancer.



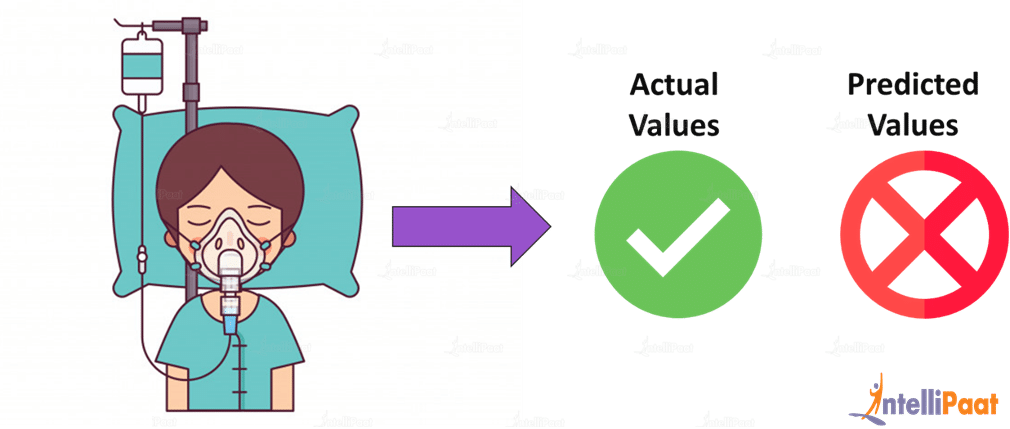
### ****True Positive****

True positive is nothing but the case where the actual value, as well as the predicted value, are true. The patient has been diagnosed with cancer, and the model also predicted that the patient had cancer.



### **False Negative:**

In false negative, the actual value is true, but the predicted value is false, which means that the patient has cancer, but the model predicted that the patient did not have cancer.



**Get 100% Hike!**

Master Most in Demand Skills Now !

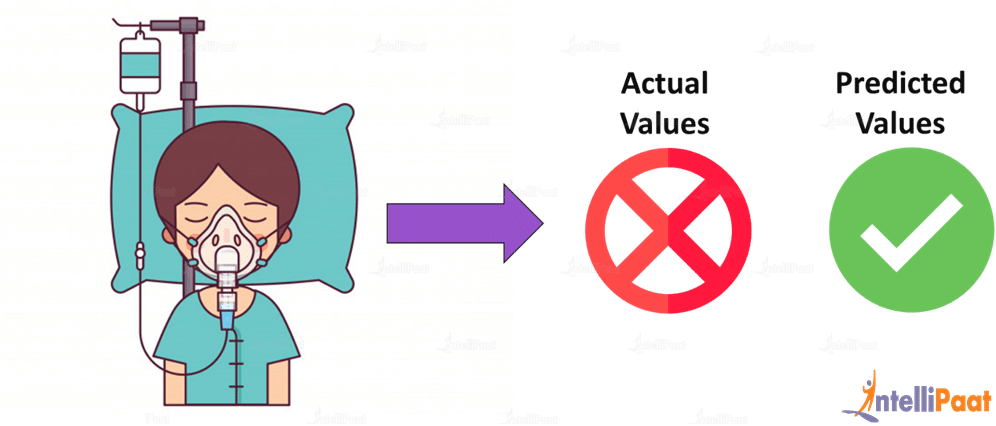
Top of Form





Bottom of Form

### **False Positive:**

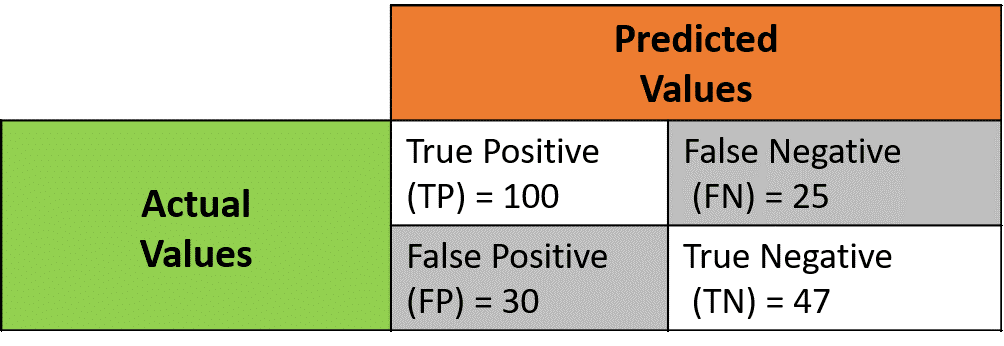
This is the case where the predicted value is true, but the actual value is false. Here, the model predicted that the patient had cancer, but in reality, the patient doesn’t have cancer. This is also known as **Type 1 Error**.  


### **True Negative:**

This is the case where the actual value is false and the predicted value is also false. In other words, the patient is not diagnosed with cancer and our model predicted that the patient did not have cancer.

## **Understanding Various Performance Metrics:**

We will be taking the help of a confusion matrix given below in order to find various performance metrics.

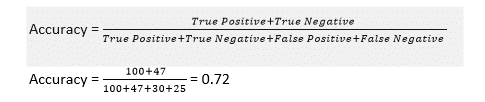


Alright, let us start with accuracy:

Accuracy or Classification Accuracy:

What: In classification problems, ‘accuracy’ refers to the number of correct predictions made by the predictive model over the rest of the predictions.

How:



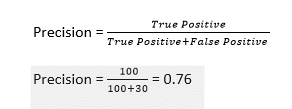
When to use: When the target variable classes in the data are nearly balanced

When not to use: When the target variables in the data are majority of one class.

Precision:

What: Here, ‘precision’ means what proportion of all predictions that we made with our predictive model is actually true.

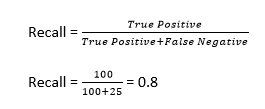
How:



It means, when our model predicts that a patient does not have cancer, it is correct 76 percent of the time.

Recall or Sensitivity:

What: ‘Recall’ is nothing but the measure that tells what proportion of patients that actually had cancer were also predicted of having cancer. It answers the question, “How sensitive the classifier is in detecting positive instances?”

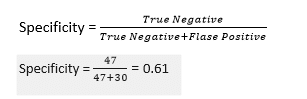


It means that 80 percent of all cancer patients are correctly predicted by the model to have cancer.

Specificity:

 It answers question, “How specific or selective is the classifier in predicting positive instances?”

How:

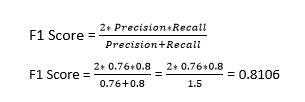


A specificity of 0.61 means 61 percent of all patients that didn’t have cancer are predicted correctly.

F1 Score:

This is nothing but the harmonic mean of precision and recall.

How:



F1 score is high, i.e., both precision and recall of the classifier indicate good results.

Created a heatmap to visualize the correlation between each numeric variable.

Sns.countplot has been implemented.

IQR and Percentile method has been included here.

## 