**BLOG SUBMISSION**

PROJECT NAME: MUSHROOM

* **PROBLEM DEFINITION:**

In this project, we will examine the data and build different **machine learning models** that will detect if the mushroom is **edible or poisonous** by its specifications like cap shape, cap color, gill color, etc. using different classifiers.

DATASET:

The dataset used in this project is mushrooms.csv that contains 8124 rows of mushrooms with 23 features like cap-shape, cap-surface, cap-color, bruises,gill-size etc.

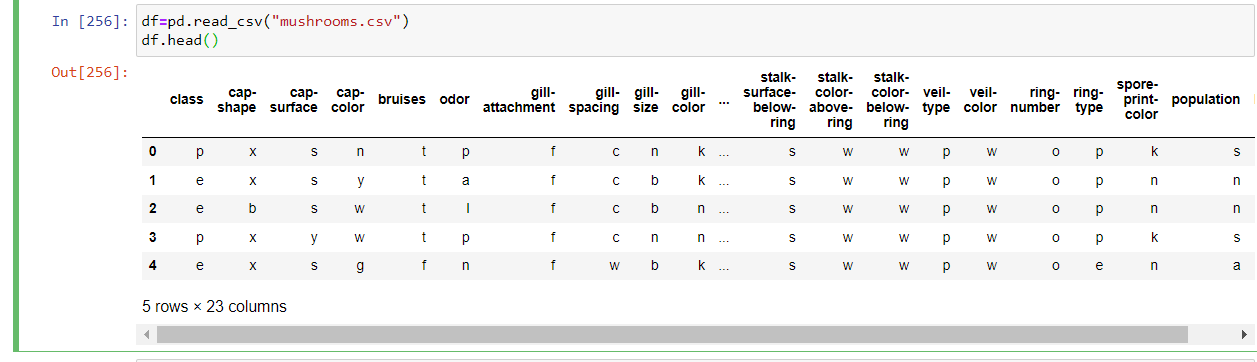
We will use these columns to classify wheather the mushrooms are edible or poisonous.

IMPORTING THE LIBRARIES and PACKAGES:



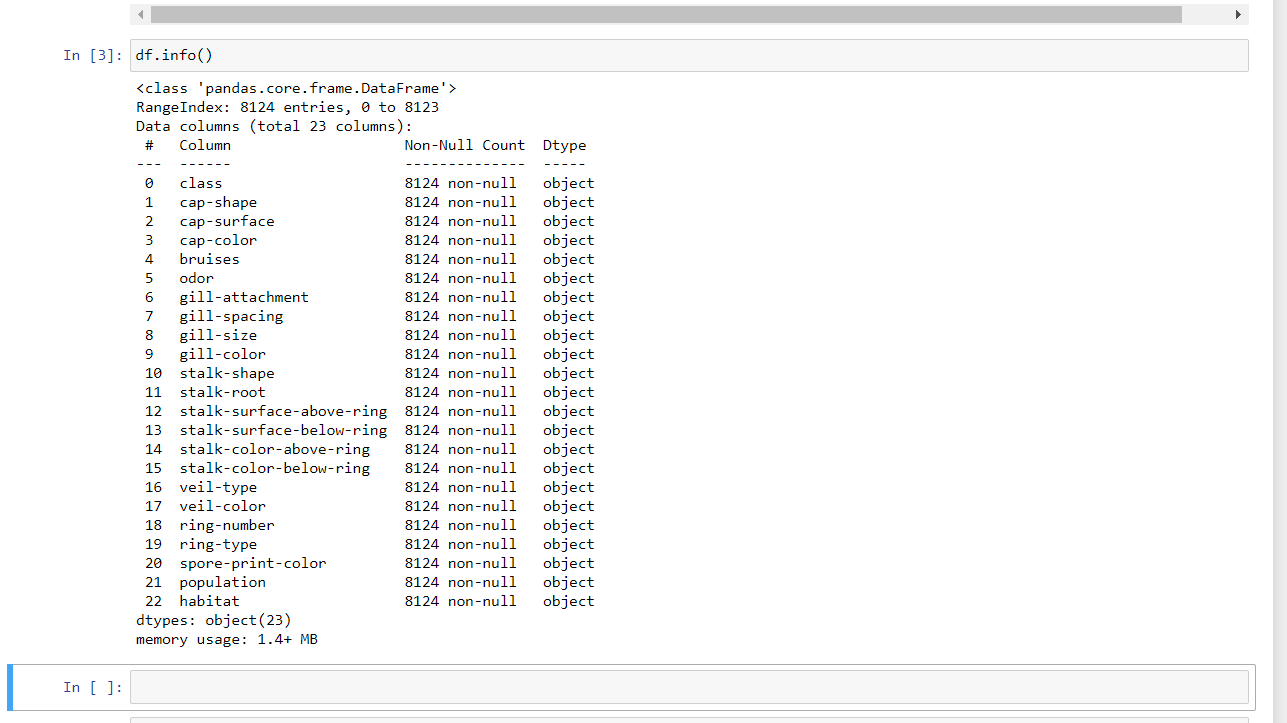
GETTING THE DATA:

Reading the csv file of the dataset

. 

* **DATA EXPLORATION/ANALYSIS:**

The **.info()** method will give you a **concise summary** of the DataFrame. This method will print the information about the DataFrame including the index dtype and column dtypes, non-null values, and memory usage. Here is the output:



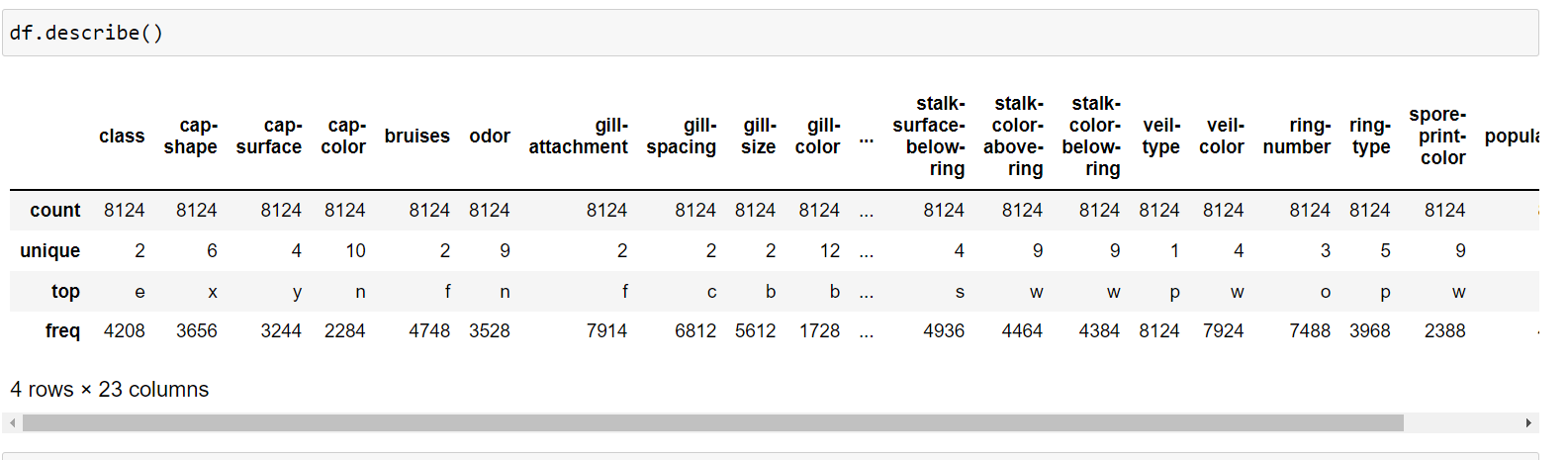
The training-set has 8124 examples and 22 features + the target variable (class). All the columns are object type.

DESCRIPTIVE STATISTICS:

The **.describe()** method will give you the **statistics** of the columns.

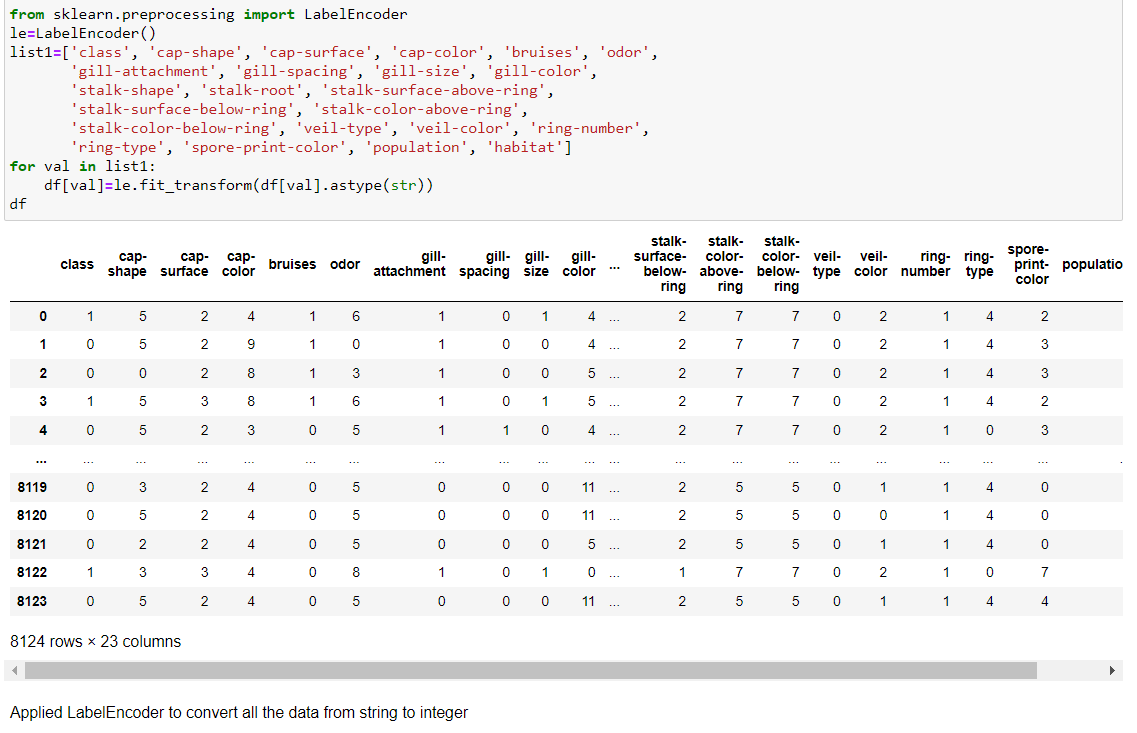
* **count** shows the number of responses.
* **unique** shows the number of unique categorical values.
* **top** shows the highest-occurring categorical value.
* **freq** shows the frequency/count of the highest-occurring categorical value.

Here is the output:



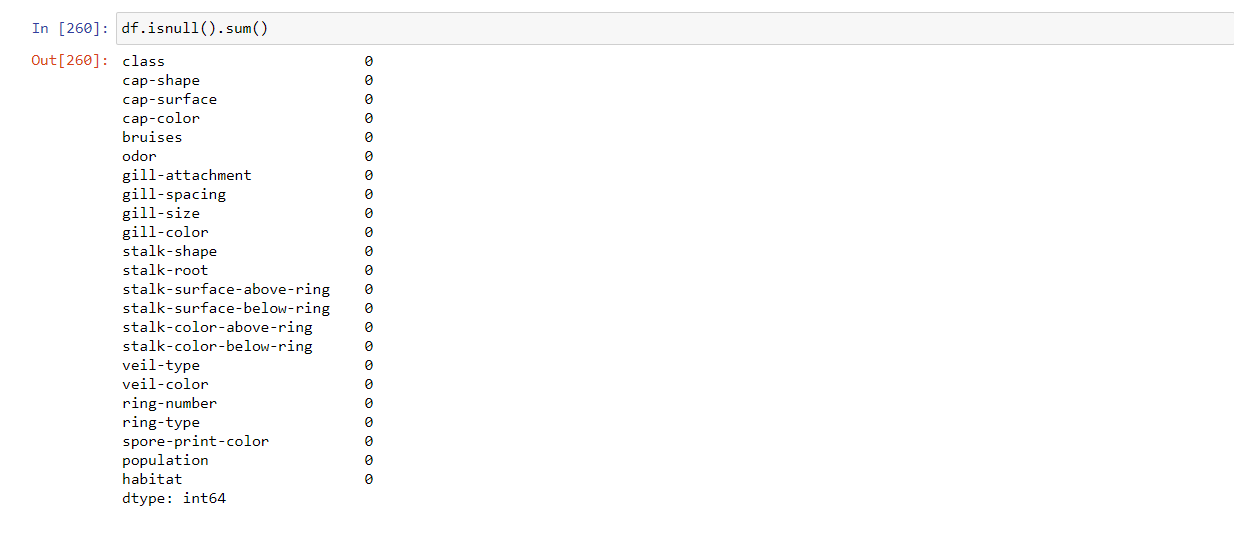
DATA MANIPULATION :

The data is categorical so we’ll use **LabelEncoder** to convert it to ordinal. **LabelEncoder** converts each value in a column to a number.



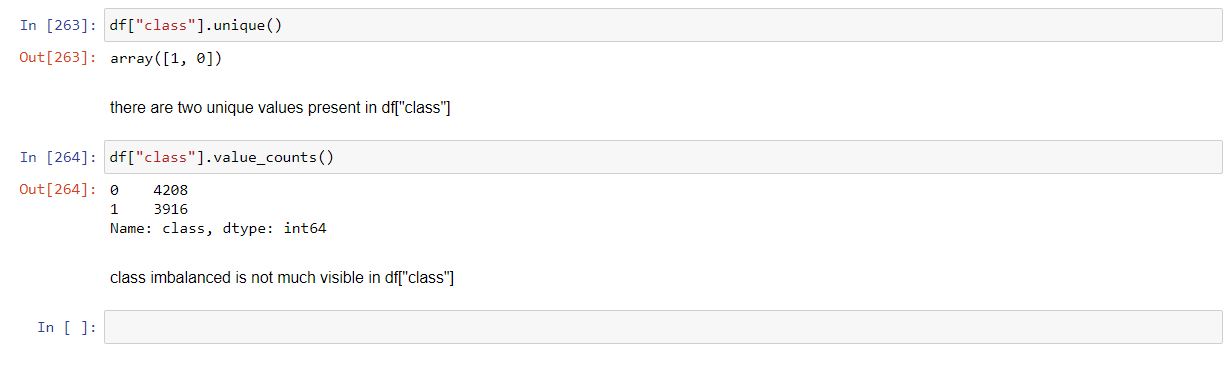
Now we see that all the column values are converted to ordinal and there are no categorical values left.

CHECK THE NULL VALUES:



There is no null values to be treated.

Count of the unique occurrences of ‘class’ column:

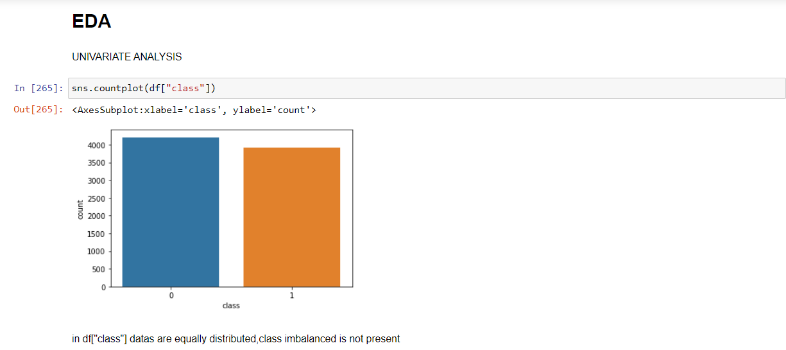


The **.value\_counts()** method will give you the count of the unique occurrences. As we can see, there are **4208 occurrences of 0 mushrooms**and**3916 occurrences of 1 mushrooms**in the dataset.

* **EDA CONCLUDING REMARKS:**

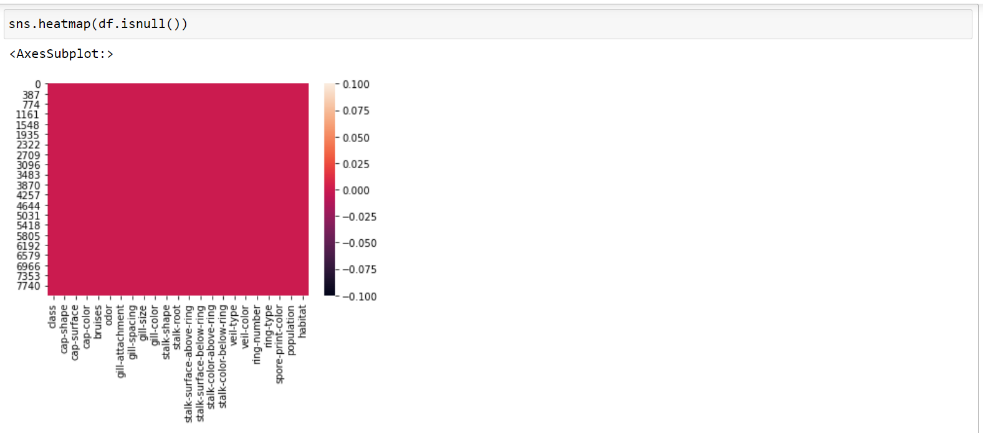
TO CHECK THE DATA IMBALANCE:

Visualization of the count of edible and poisonous mushrooms using Seaborn



Here we can clearly, that in class column datas are balanced.

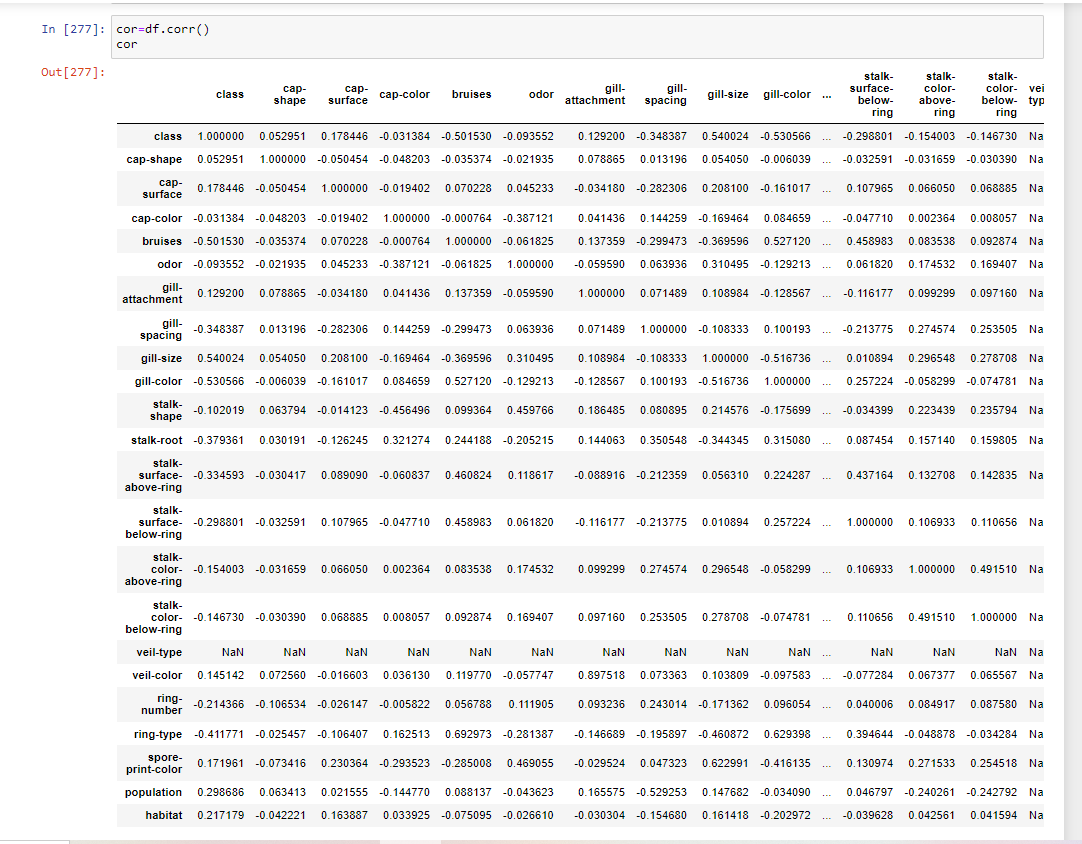
To check the null values:

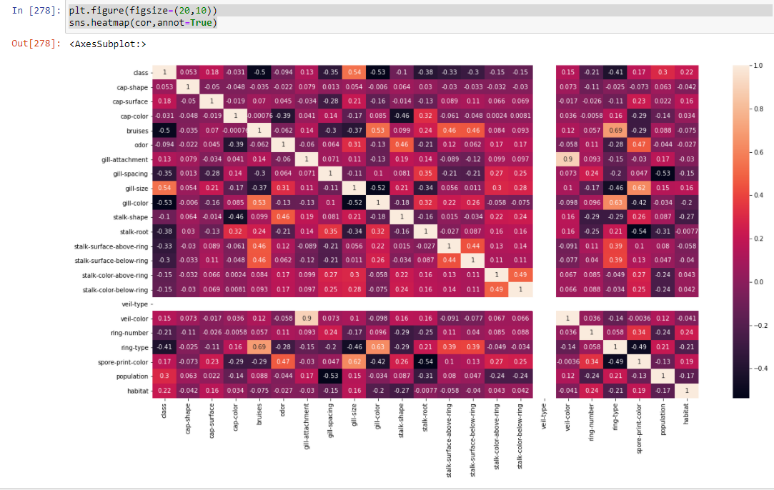


As we can see from the heatmap there is no null values present in the dataset.

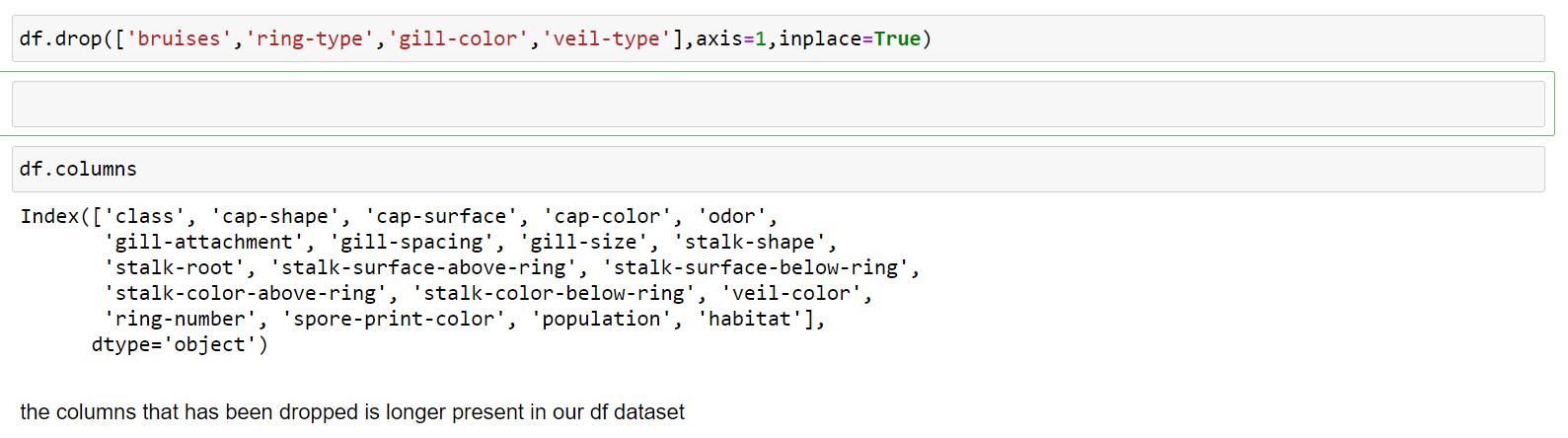
* **PRE-PROCESSING PIPELINE:**

TO CHECK THE CORRELATION:



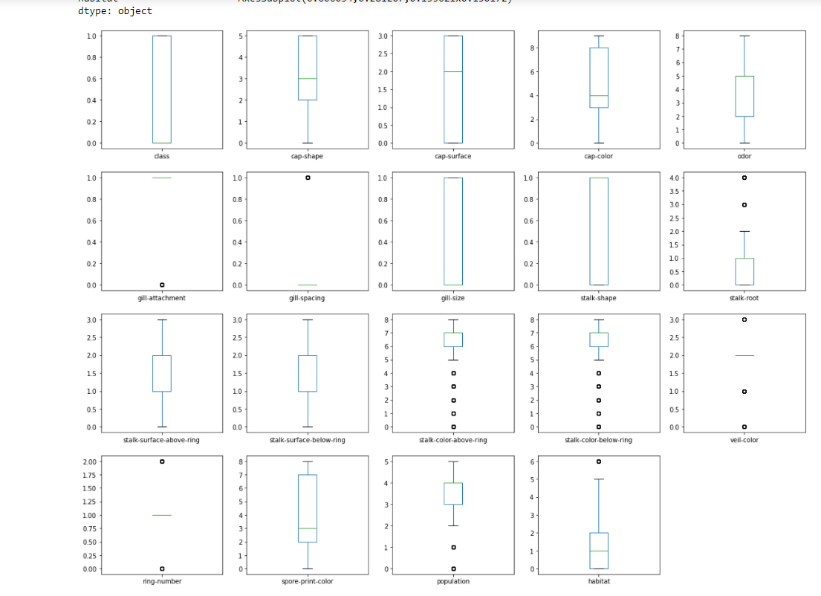


There are few columns('bruises','gill-spacing','gill-color','stalk-root','stalk-surface-above-ring','ring-type') that are negatively correated with our target variable(y) class. Hence we will be dropping these. Also, the column “veil-type” is 0 and not contributing to the data so we’ll remove it.

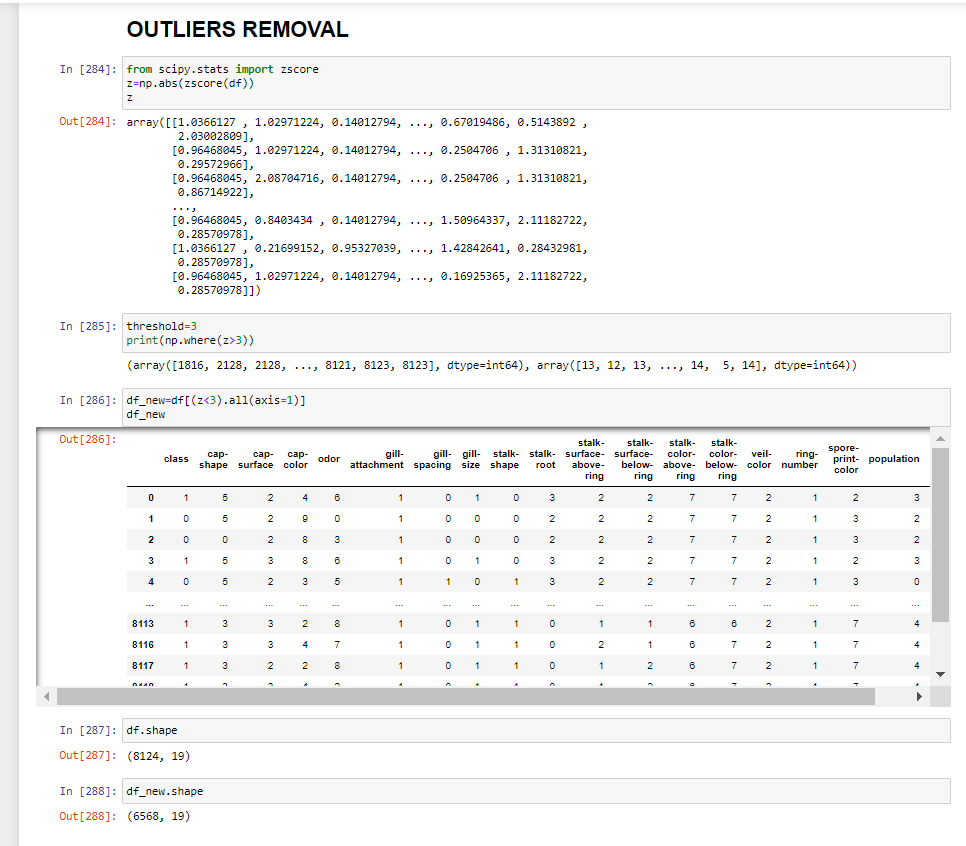


By using **df.columns** we checked, dropped columns are no longer present in the dataset.

CHECKING THE OUTLIERS:

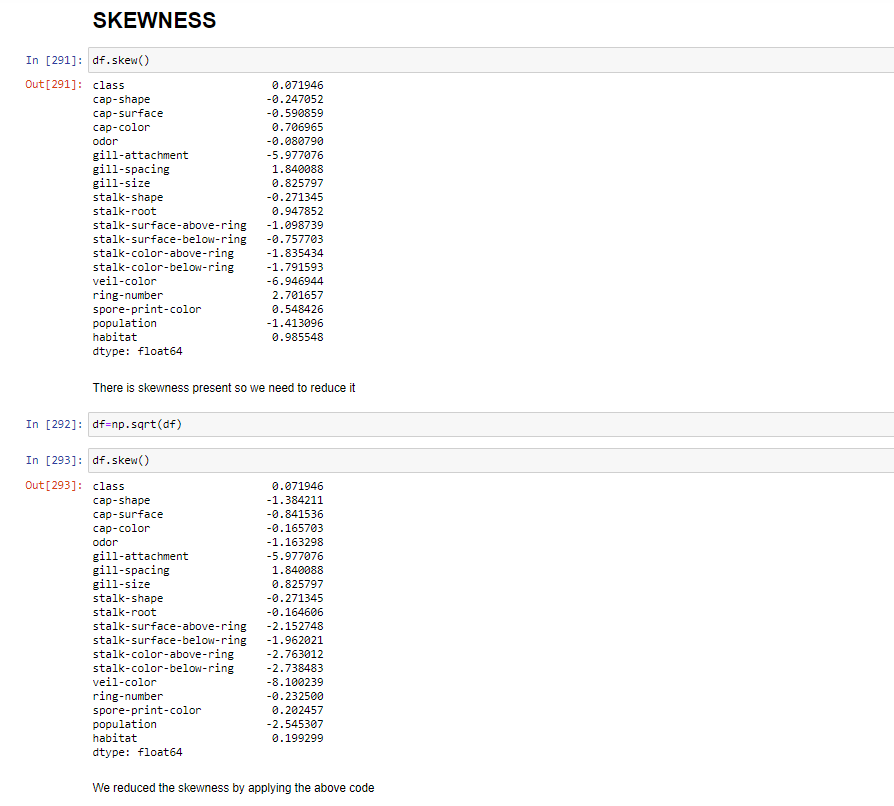


There are outliers present in columns as such "gill-attachment","gill-spacing","stalk-root","stalk-surface-above-ring","stalk-color-below-ring","ring-number","population","habitat". So we need to remove it



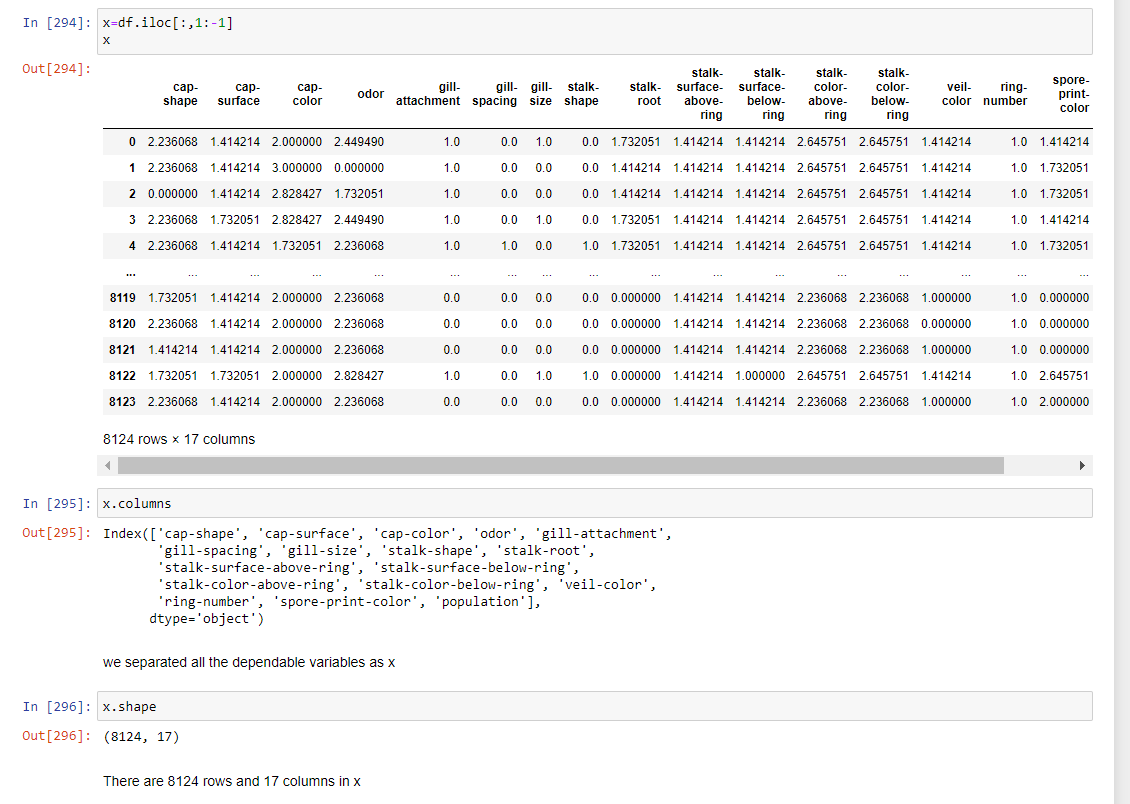
Outliers has been removed now.

REMOVING the SKEWNESS:



As we can see skewness has been removed now.

Separating the feature and target column:

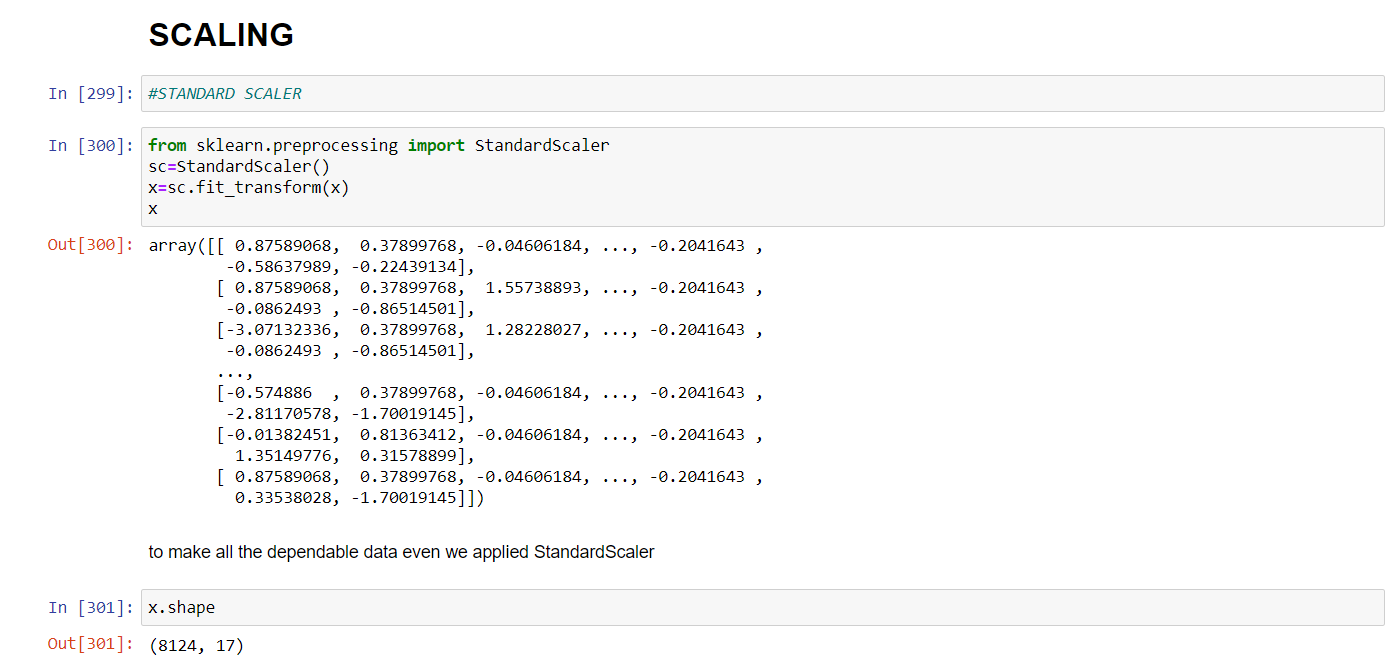


We separated the feature column as x



We separated the target column as y

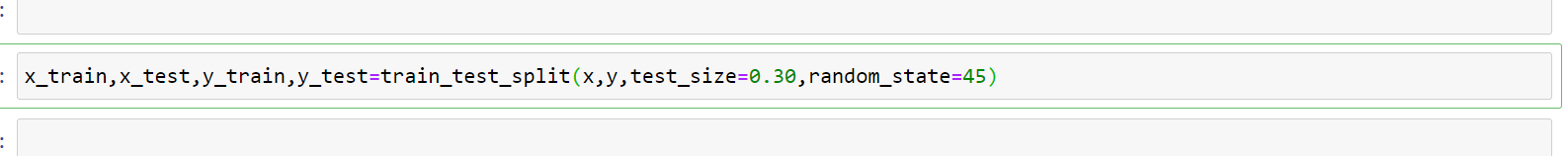
Applied StandardScaler in feature column to make all the data into same scale



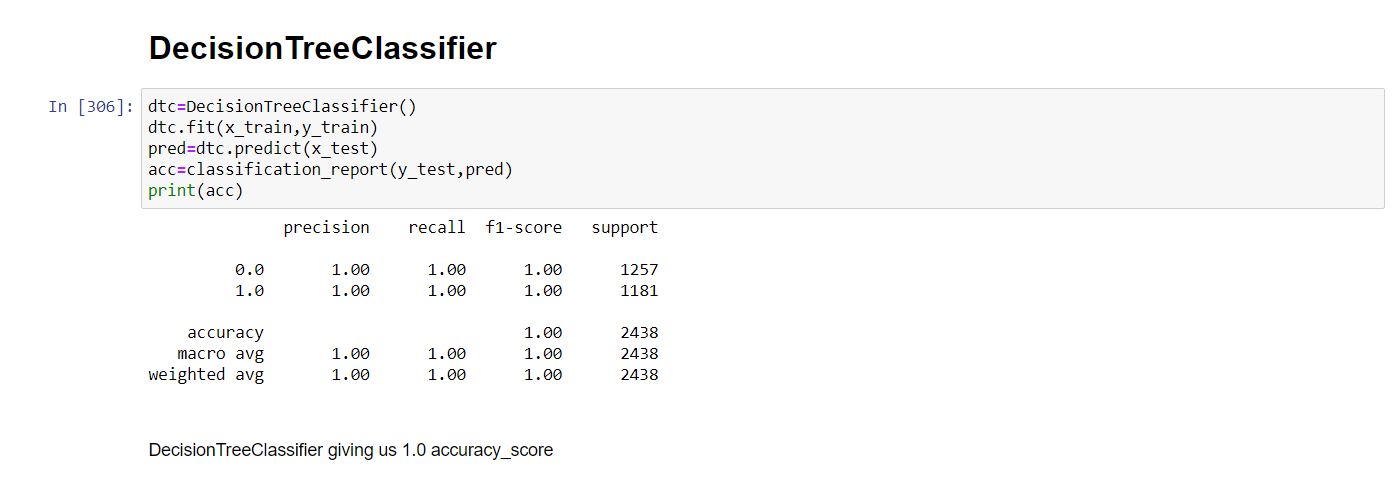
* **BUILDING MACHINE LEARNING MODELS:**

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

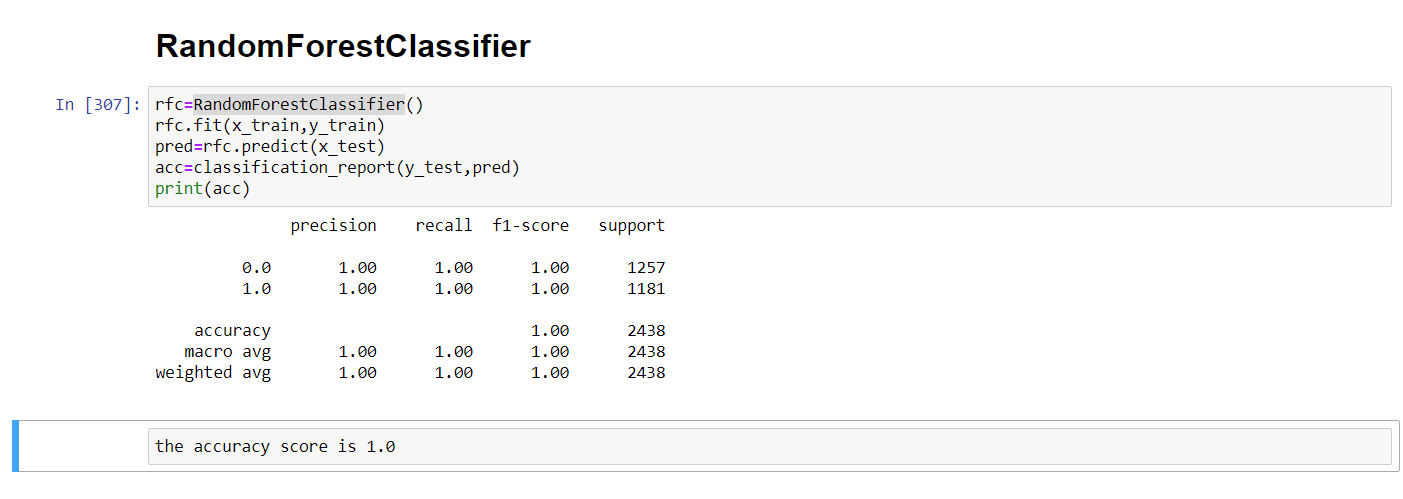
Classification methods:



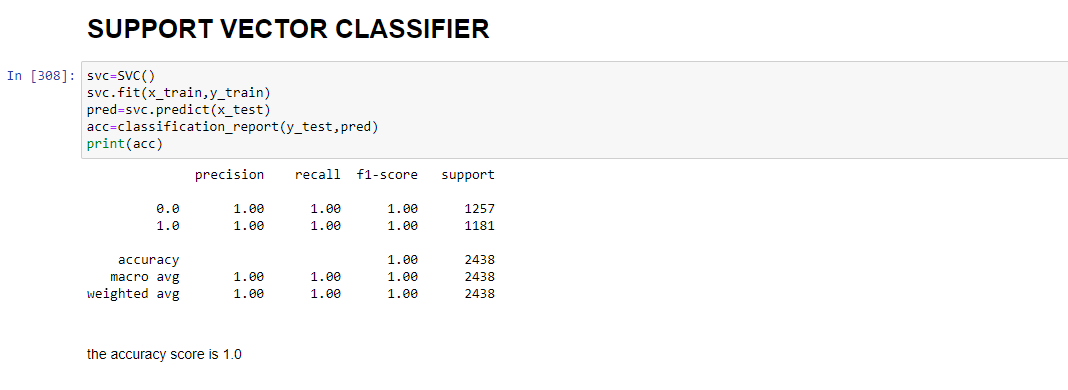
DecisionTreeClassifier:



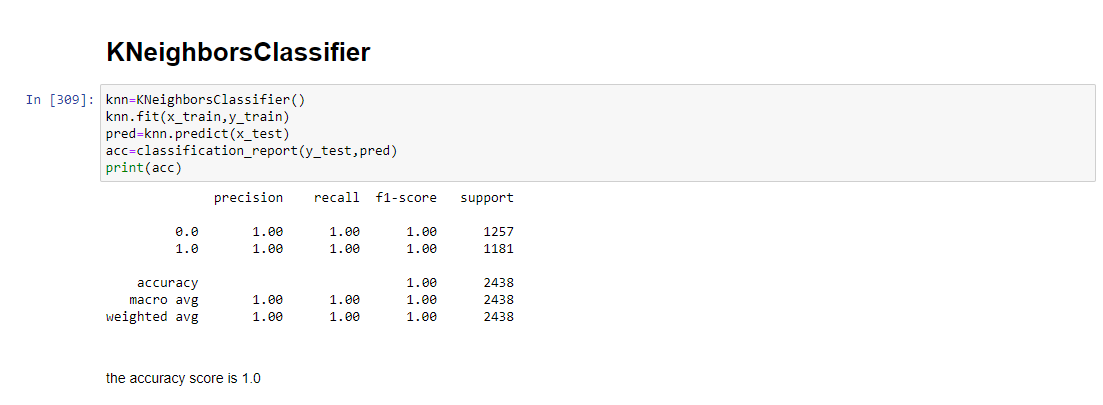
RandomForestClassifier:



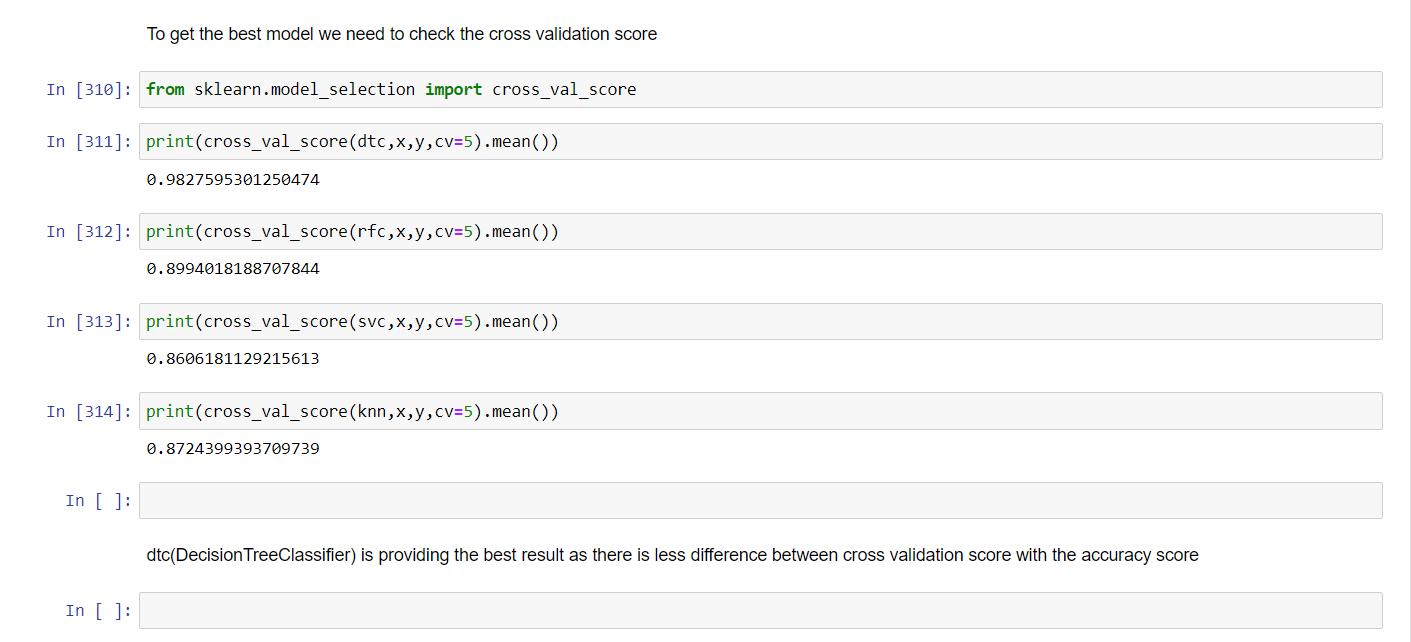
SupportVectorClassifier:



KNeighborsClassifier:



CROSS VALIDATION SCORE:



As we can see by checking the cross validation score DecisionTreeClassifier

Is giving the best score.

Confusion matrix using DecisionTreeClassifier:

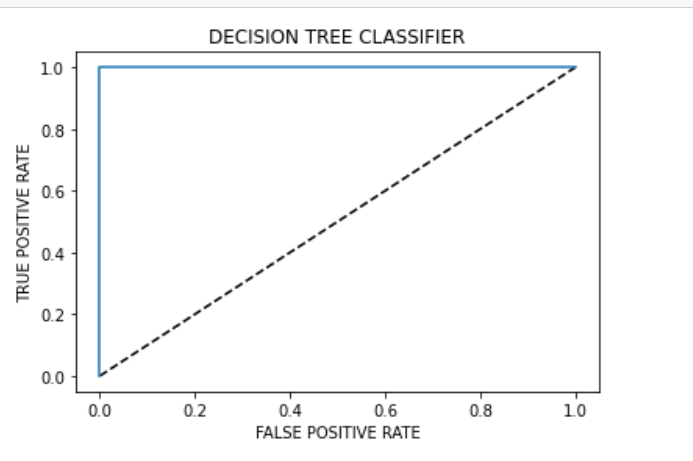


Building the AUC ROC Curve:

Another way to evaluate and compare your binary classifier is provided by the ROC AUC Curve.



Here is the output:



The black line in the middel represents a purely random classifier and therefore your classifier should be as far away from it as possible. Our **DecisionTreeClassifier** model is doing the best job.

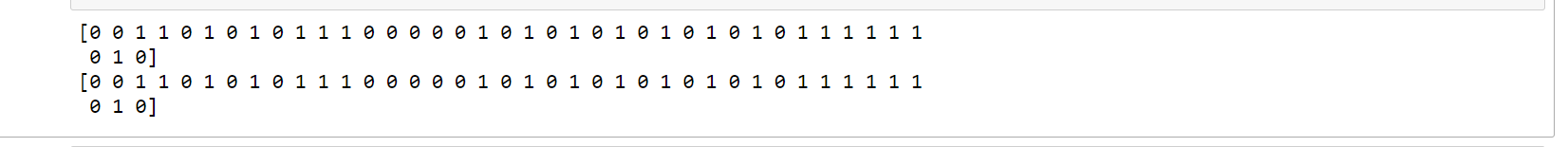
* **CONCLUDING REMARKS:**

Predictions:

Predicting some of the x\_test results and matching it with true i.e. y\_test values using Decision Tree Classifier.



Here is the Output:



As we can see, the **predicted and the true values match 100%.**

**Conclusion:**

From the **confusion matrix**, we saw that our **train and test data is balanced.**