International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 10, ISSN No. 2455-2143, Pages 110-116
Published Online February 2023 in IJEAST (http://www.ijeast.com)

PRACTICAL CLASSIFICATION TEMPLATE FOR DATASETS IN MACHINE LEARNING

Syed Mohammed Idris
Department of Information Science and Engineering
HKBK College of Engineering
Bangalore, Karnataka, India

Abstract— **In this work, different Machine Learning (ML) algorithms are used and evaluated based on their performance of classifying peer reviewed content of the dataset provided. The ultimate objective is to extract meaningful information from the classification of the given dataset. In pursuing this objective, the ML techniques are utilized to classify different datasets into: Validation Dataset and Test Dataset. The ML techniques applied in this work are Logistic Regression, Support Vector Machines, Naïve Bayes, Linear Discriminant Analysis, K-Nearest Neighbor, and Decision Tree. In addition to the description of the utilized ML algorithms, the methodology and algorithms for classification using the aforementioned ML techniques are provided. The comparative study based on six different performance measures suggests that - with the exception of Support Vector Machines algorithm - the proposed ML techniques with the detailed pre-processing algorithms may or may not work well for classifying the iris flower dataset.**

Keywords— **Machine Learning, classification, dataset, iris flower**

I. INTRODUCTION

The problem of the classification of various datasets is of remarkable importance for these services, since it enables increased functionality and improved performance. For instance, a robust classification strategy allows the user to perform searches by focusing on only a specific portion of the indexed documents, thus increasing both effectiveness and efficiency. Additional potential benefits include similar documents recommendations, collaborative filtering, query expansion facilities, expert identification, and so on. Several methods have been proposed to address the issue of identifying the research field a scientific article belongs to. These methods include keyword extraction algorithms which attempt to identify repeated textual patterns and extract the most representative keywords from the article. In the sequel, they employ traditional classification approaches such as Support Vector Machines (SVM) to identify the research field that best describes the content of the article.

In this paper, we are proposing how Machine learning approach is a technique used to teach machines how to handle the data more efficiently and get result more accurately. In Some cases after viewing the data, we cannot understand the pattern or extract information from the data. In such case, we apply machine learning techniques for predicate the data. Large quantity of datasets are available from different sources, there is a demand for machine learning. Many industries from medicine to military are applying machine learning to extract relevant information from the available datasets. The main purpose of machine learning is to learn from the existing data. Large set of algorithms are design how to make machines learn by themselves. Many mathematicians and programmers apply several approaches to find the solution of this problem.

A comparative analysis is done in this paper, based on text classification employing Machine Learning techniques on different datasets. The problem is that the manual process of classifying text data is tedious and very time-consuming. Therefore, it is very important to automate the process and enhance data-driven decisions. In this research, the machine learning algorithms are applied and compared for best performance on different datasets. The documents in the text classification model are passed through different steps viz (i) convert the main document into plain text, lowercase the full document, removal of stop words, remove the words which are not useful, to reduce different words in a single word called root word using stemming and lemmatization and(ii) select the data for training and testing, build the classifier and then deploy the classifier on different datasets. Further, the Machine learning techniques can also be applied for classification problems to measure the perceptions about the COVID-19 pandemic and identify the misconception among the population, which will help us to inform the public health organizations and then better methods can be created to educate the public and make them understand not to fall for these misconceptions. Machine learning techniques can also be used to tackle the SARAS-COV-2 crisis in the field of diagnosis, disease progression, and epidemiology.

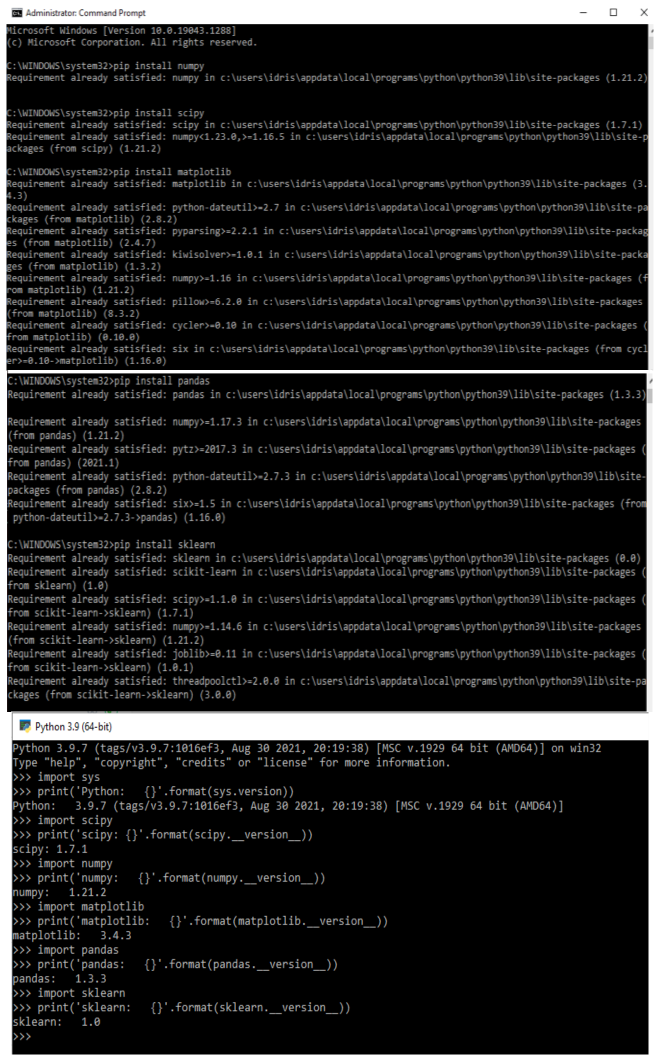
Nowadays, any news or information spreads swiftly on many social networking sites, and there is no way of knowing whether it is authentic or true, even if we trust it. Most individuals are using this platform as a weapon to manipulate public opinion for political, religious, or other causes, but we can use machine learning algorithms to determine whether the news or information is authentic or deceptive propaganda. A conventional way of extracting the emotions, opinions, and

attitudes from the text data available on social networking sites can also be done using machine learning algorithms. Different data preprocessing techniques are also essential for models to give better results with good performance. TF-IDF is a statistical measure for extracting meaningful information from text input, but it is ineffective for unbalanced distributions. However, we may apply an upgraded version of TF-IDF to improve the model's power and robustness. There are a variety of additional ways for classifying text data, such as a caps-net based multitask learning architecture for text classification, but when compared to machine learning approaches for the same problem, the results are substantially better utilizing machine learning techniques. Following the release of new COVID-19 variations, the entire world is experiencing healthcare issues, making it harder to collect health care records to analyze current and future needs for the general public using various machine learning approaches
Machine learning is one of the most important technical developments allowing Industry 4.0 to take hold in businesses and industries. The introduction of AI and Machine Learning to Industry 4.0 marks a significant shift for manufacturing organizations, potentially resulting in new business prospects and benefits such as increased productivity. Artificial Intelligence, Machine Learning, Deep Learning have also been widely used in areas like healthcare, finance, smart factories as a part of Industry 4.0. The reader should be cautioned that a single article cannot be a comprehensive review of all classification learning algorithms. Instead, our goal has been to provide a representative sample of existing lines of research in each learning technique. In each of our listed areas, there are many other papers that more comprehensively detail relevant work.

II. EXPERIMENT

A. Load/Import and the versions of the libraries –

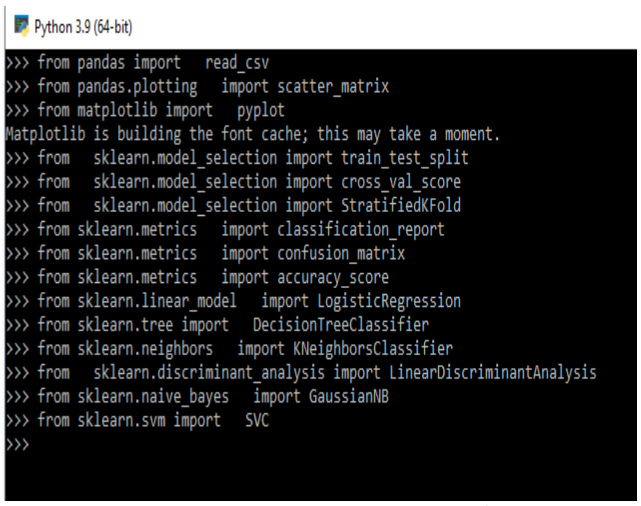
SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation. **Scikit-learn** is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML. **Pandas** is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data. **Matplotlib** is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc. **NumPy** is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.



```
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>pip install numpy
Requirement already satisfied: numpy in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (1.21.2)


C:\WINDOWS\system32>pip install scipy
Requirement already satisfied: scipy in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (1.7.1)
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in c:\users\idris\appdata\local\programs\python\python39\lib\site-p
ackages (from scipy) (1.21.2)

C:\WINDOWS\system32>pip install matplotlib
Requirement already satisfied: matplotlib in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (3.
4.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\idris\appdata\local\programs\python\python39\lib\site-pa
ckages (from matplotlib) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packag
es (from matplotlib) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packa
ges (from matplotlib) (1.3.2)
Requirement already satisfied: numpy>=1.16 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (f
rom matplotlib) (1.21.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages
(from matplotlib) (8.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (
from matplotlib) (0.10.0)
Requirement already satisfied: six in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (from cycl
er>=0.10->matplotlib) (1.16.0)
```

```
C:\WINDOWS\system32>pip install pandas
Requirement already satisfied: pandas in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (1.3.3)

Requirement already satisfied: numpy>=1.17.3 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages
(from pandas) (1.21.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (
from pandas) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\idris\appdata\local\programs\python\python39\lib\site-
packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (from
 python-dateutil>=2.7.3->pandas) (1.16.0)

C:\WINDOWS\system32>pip install sklearn
Requirement already satisfied: sklearn in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (
from sklearn) (1.0)
Requirement already satisfied: scipy>=1.1.0 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (
from scikit-learn->sklearn) (1.7.1)
Requirement already satisfied: numpy>=1.14.6 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages
(from scikit-learn->sklearn) (1.21.2)
Requirement already satisfied: joblib>=0.11 in c:\users\idris\appdata\local\programs\python\python39\lib\site-packages (
from scikit-learn->sklearn) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\idris\appdata\local\programs\python\python39\lib\site-pa
ckages (from scikit-learn->sklearn) (3.0.0)
```

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> print('Python:   {}'.format(sys.version))
Python:   3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)]
>>> import scipy
>>> print('scipy: {}'.format(scipy.__version__))
scipy: 1.7.1
>>> import numpy
>>> print('numpy:   {}'.format(numpy.__version__))
numpy:   1.21.2
>>> import matplotlib
>>> print('matplotlib:   {}'.format(matplotlib.__version__))
matplotlib:   3.4.3
>>> import pandas
>>> print('pandas:   {}'.format(pandas.__version__))
pandas:   1.3.3
>>> import sklearn
>>> print('sklearn:   {}'.format(sklearn.__version__))
sklearn:   1.0
>>>
```

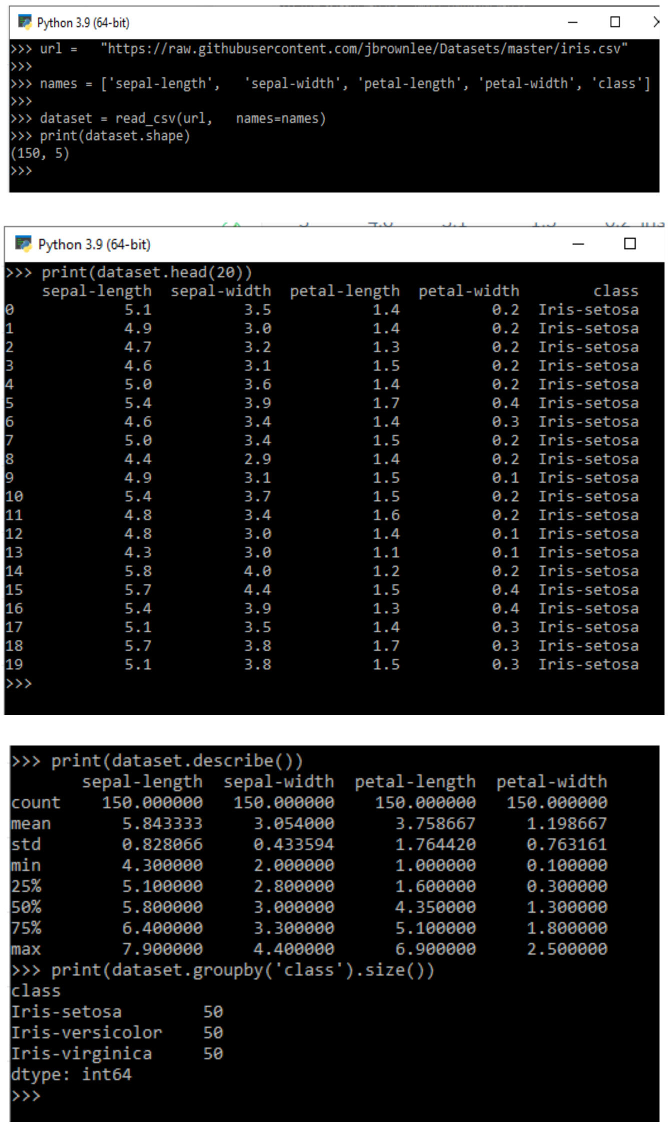


```
>>> from pandas import   read_csv
>>> from pandas.plotting   import scatter_matrix
>>> from matplotlib import   pyplot
Matplotlib is building the font cache; this may take a moment.
>>> from   sklearn.model_selection import train_test_split
>>> from   sklearn.model_selection import cross_val_score
>>> from   sklearn.model_selection import StratifiedKFold
>>> from sklearn.metrics   import classification_report
>>> from sklearn.metrics   import confusion_matrix
>>> from sklearn.metrics   import accuracy_score
>>> from sklearn.linear_model   import LogisticRegression
>>> from sklearn.tree import   DecisionTreeClassifier
>>> from sklearn.neighbors   import KNeighborsClassifier
>>> from   sklearn.discriminant_analysis import LinearDiscriminantAnalysis
>>> from sklearn.naive_bayes   import GaussianNB
>>> from sklearn.svm import   SVC
>>>
```

Fig. 1. Importing and Checking version of the libraries

B. Loading the dataset and its representations–

CSV (comma-separated value) files are a common file format for transferring and storing data. The ability to read, manipulate, and write data to and from CSV files using Python is a key skill to master for any data scientist or business analysis. In this post, we'll go over what CSV files are, how to read CSV files into Pandas Data Frames, and how to write Data Frames back to CSV files post analysis. The basic process of loading data from a CSV file into a Pandas Data Frame (with all going well) is achieved using the "read_csv" function in Pandas. Pandas is the most popular data manipulation package in Python, and Data Frames are the Pandas data type for storing tabular 2D data. A "CSV" file, that is, a file with a "csv" filetype, is a basic text file. Any text editor such as Note Pad on windows or TextEdit on Mac, can open a CSV file and show the contents. Sublime Text is a wonderful and multi-functional text editor option for any platform.CSV is a standard for storing tabular data in text format, where commas are used to separate the different columns, and newlines (carriage return / press enter) used to separate rows. Typically, the first row in a CSV file contains the names of the columns for the data. Note that almost any tabular data can be stored in CSV format – the format is popular because of its simplicity and flexibility. You can create a text file in a text editor, save it with a .csv extension, and open that file in Excel or Google Sheets to see the table form. As mentioned before, CSV files do not contain any type information for data. Data types are inferred through examination of the top rows of the file, which can lead to errors. To manually specify the data types for different columns, the dtype parameter can be used with a dictionary of column names and data types to be applied.



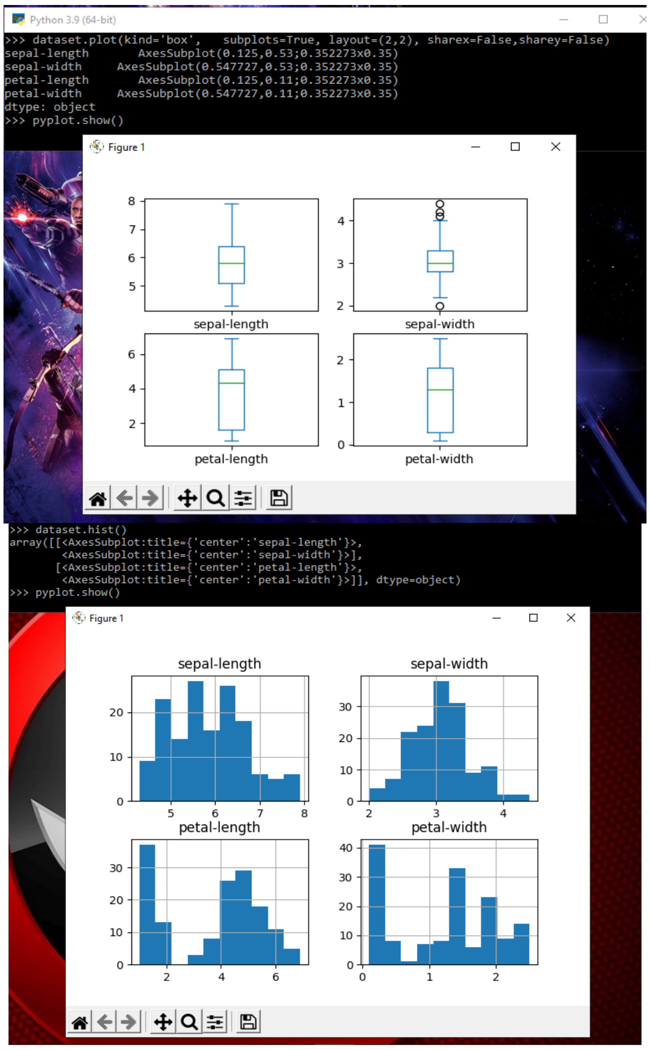
```
>>> url =   "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
>>>
>>> names = ['sepal-length',   'sepal-width', 'petal-length', 'petal-width', 'class']
>>>
>>> dataset = read_csv(url,   names=names)
>>> print(dataset.shape)
(150, 5)
>>>
```

```
>>> print(dataset.head(20))
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
>>>
```

```
>>> print(dataset.describe())
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
>>> print(dataset.groupby('class').size())
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
>>>
```

Fig. 2. Loading the dataset and its distributions

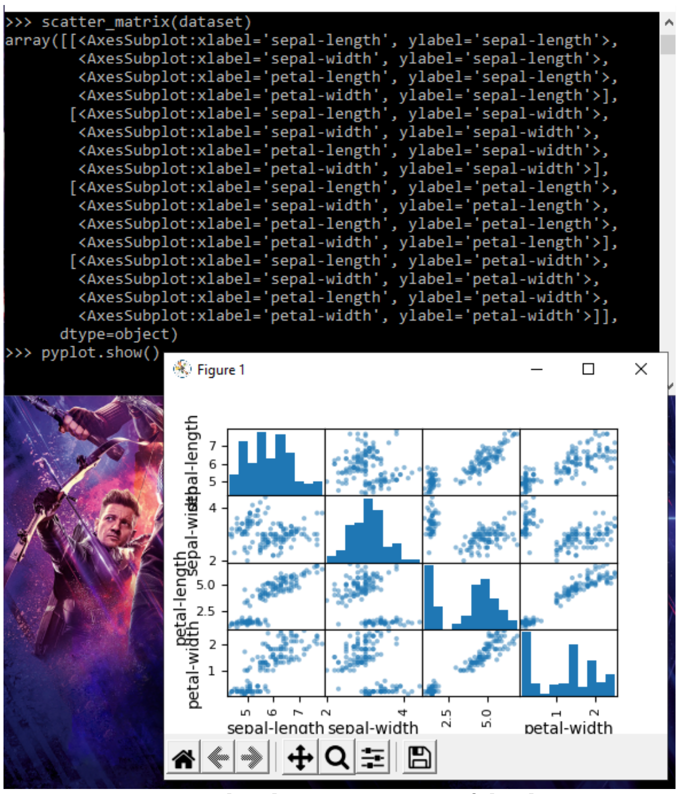
Data visualization is a field in data analysis that deals with visual representation of data. It graphically plots data and is an effective way to communicate inferences from data. Using data visualization, we can get a visual summary of our data. With pictures, maps and graphs, the human mind has an easier time processing and understanding any given data. Data visualization plays a significant role in the representation of both small and large data sets, but it is especially useful when we have large data sets, in which it is impossible to see all of our data, let alone process and understand it manually. Python offers several plotting libraries, namely Matplotlib, Seaborn and many other such data visualization packages with different features for creating informative, customized, and appealing plots to present data in the most simple and effective way.Matplotlib and Seaborn are python libraries that are used for data visualization. They have inbuilt modules for plotting different graphs. While Matplotlib is used to embed graphs into applications, Seaborn is primarily used for statistical graphs. Scatter plots are used when we have to plot two or more variables present at different coordinates. The data is scattered all over the graph and is not confined to a range. Two or more variables are plotted in a Scatter Plot, with each variable being represented by a different color.



```
>>> dataset.plot(kind='box',   subplots=True, layout=(2,2), sharex=False,sharey=False)
sepal-length       AxesSubplot(0.125,0.53;0.352273x0.35)
sepal-width     AxesSubplot(0.547727,0.53;0.352273x0.35)
petal-length       AxesSubplot(0.125,0.11;0.352273x0.35)
petal-width     AxesSubplot(0.547727,0.11;0.352273x0.35)
dtype: object
>>> pyplot.show()
```

```
>>> dataset.hist()
array([[<AxesSubplot:title={'center':'sepal-length'}>,
        <AxesSubplot:title={'center':'sepal-width'}>],
       [<AxesSubplot:title={'center':'petal-length'}>,
        <AxesSubplot:title={'center':'petal-width'}>]], dtype=object)
>>> pyplot.show()
```

A Histogram is a bar representation of data that varies over a range. It plots the height of the data belonging to a range along the y-axis and the range along the x-axis. Histograms are used to plot data over a range of values. They use a bar representation to show the data belonging to each range. Let's again use the 'Iris' data which contains information about flowers to plot histograms.



```
>>> scatter_matrix(dataset)
array([[<AxesSubplot:xlabel='sepal-length', ylabel='sepal-length'>,
        <AxesSubplot:xlabel='sepal-width', ylabel='sepal-length'>,
        <AxesSubplot:xlabel='petal-length', ylabel='sepal-length'>,
        <AxesSubplot:xlabel='petal-width', ylabel='sepal-length'>],
       [<AxesSubplot:xlabel='sepal-length', ylabel='sepal-width'>,
        <AxesSubplot:xlabel='sepal-width', ylabel='sepal-width'>,
        <AxesSubplot:xlabel='petal-length', ylabel='sepal-width'>,
        <AxesSubplot:xlabel='petal-width', ylabel='sepal-width'>],
       [<AxesSubplot:xlabel='sepal-length', ylabel='petal-length'>,
        <AxesSubplot:xlabel='sepal-width', ylabel='petal-length'>,
        <AxesSubplot:xlabel='petal-length', ylabel='petal-length'>,
        <AxesSubplot:xlabel='petal-width', ylabel='petal-length'>],
       [<AxesSubplot:xlabel='sepal-length', ylabel='petal-width'>,
        <AxesSubplot:xlabel='sepal-width', ylabel='petal-width'>,
        <AxesSubplot:xlabel='petal-length', ylabel='petal-width'>,
        <AxesSubplot:xlabel='petal-width', ylabel='petal-width'>]],
      dtype=object)
>>> pyplot.show()
```

Fig. 3. Graphical representations of the dataset

III. IMPLEMENTATION AND RESULT

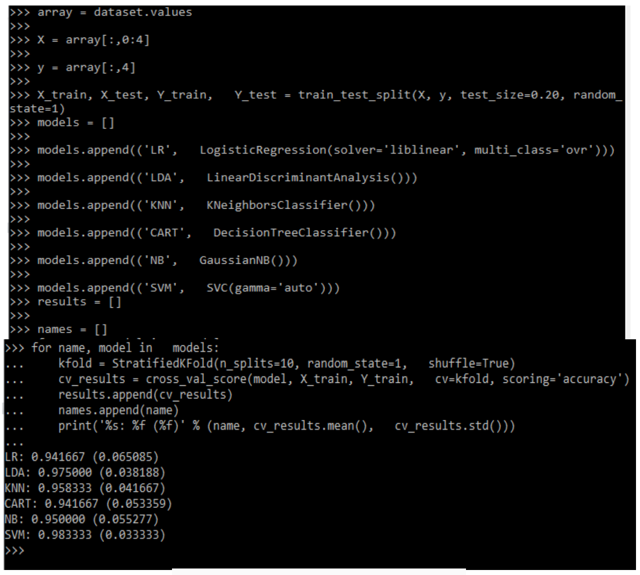
A. Implement Train-Test Split method

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced. The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset. The idea of "sufficiently large" is specific to each predictive modeling problem. It means that there is enough data to split the dataset into train and test datasets and each of the train and test datasets are suitable representations of the problem domain.

This requires that the original dataset is also a suitable representation of the problem domain. A suitable representation of the problem domain means that there are enough records to cover all common cases and most uncommon cases in the domain. This might mean combinations of input variables observed in practice. It might require thousands, hundreds of thousands, or millions of examples.

Conversely, the train-test procedure is not appropriate when the dataset available is small. The reason is that when the dataset is split into train and test sets, there will not be enough data in the training dataset for the model to learn an effective mapping of inputs to outputs. There will also not be enough data in the test set to effectively evaluate the model performance. The estimated performance could be overly optimistic (good) or overly pessimistic (bad). If you have insufficient data, then a suitable alternate model evaluation procedure would be the k-fold cross-validation procedure. In addition to dataset size, another reason to use the train-test split evaluation procedure is computational efficiency. Some models are very costly to train, and in that case, repeated evaluation used in other procedures is intractable. An example might be deep neural network models.

In this case, the train-test procedure is commonly used. Alternately, a project may have an efficient model and a vast dataset, although may require an estimate of model performance quickly. Again, the train-test split procedure is approached in this situation. Samples from the original training dataset are split into the two subsets using random selection. This is to ensure that the train and test datasets are representative of the original dataset.



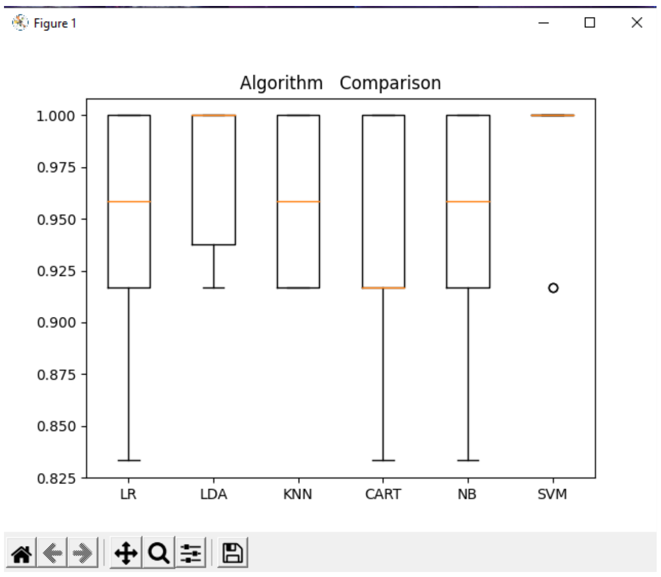
```
>>> array = dataset.values
>>>
>>> X = array[:,0:4]
>>>
>>> y = array[:,4]
>>>
>>> X_train, X_test, Y_train,   Y_test = train_test_split(X, y, test_size=0.20, random_
state=1)
>>> models = []
>>>
>>> models.append(('LR',   LogisticRegression(solver='liblinear', multi_class='ovr')))
>>>
>>> models.append(('LDA',   LinearDiscriminantAnalysis()))
>>>
>>> models.append(('KNN',   KNeighborsClassifier()))
>>>
>>> models.append(('CART',   DecisionTreeClassifier()))
>>>
>>> models.append(('NB',   GaussianNB()))
>>>
>>> models.append(('SVM',   SVC(gamma='auto')))
>>> results = []
>>>
>>> names = []
>>> for name, model in   models:
...     kfold = StratifiedKFold(n_splits=10, random_state=1,   shuffle=True)
...     cv_results = cross_val_score(model, X_train, Y_train,   cv=kfold, scoring='accuracy')
...     results.append(cv_results)
...     names.append(name)
...     print('%s: %f (%f)' % (name, cv_results.mean(),   cv_results.std()))
...
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.053359)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
>>>
```

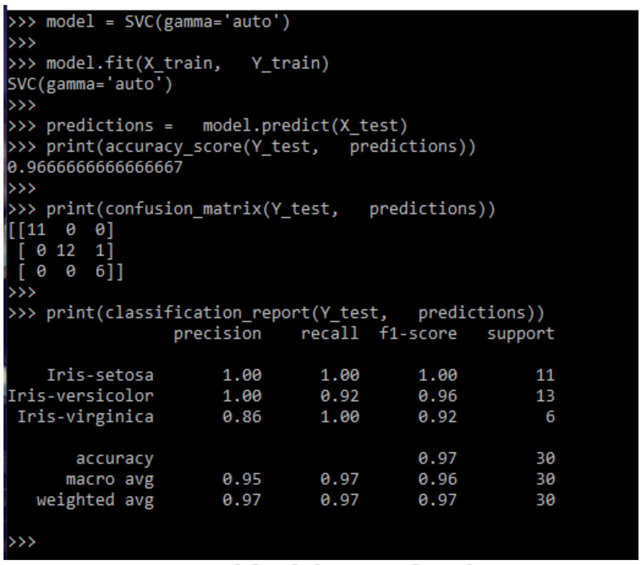
Fig. 4. Split out test dataset

B. Comparison of algorithms and making predictions on validation dataset

It is important to compare the performance of multiple different machine learning algorithms consistently. In this post you will discover how you can create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. You can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. When you work on a machine learning project, you often end up with multiple good models to choose from. Each model will have different performance characteristics.

Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. You need to be able to use these estimates to choose one or two best models from the suite of models that you have created. When you have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.





```
>>> model = SVC(gamma='auto')
>>>
>>> model.fit(X_train,   Y_train)
SVC(gamma='auto')
>>>
>>> predictions =   model.predict(X_test)
>>> print(accuracy_score(Y_test,   predictions))
0.9666666666666667
>>>
>>> print(confusion_matrix(Y_test,   predictions))
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
>>>
>>> print(classification_report(Y_test,   predictions))
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30

>>>
```

Fig. 5. Model validation and prediction

In principle, model validation is very simple: after choosing a model and its hyperparameters, we can estimate how effective it is by applying it to some of the training data and comparing the prediction to the known value. One disadvantage of using a holdout set for model validation is that we have lost a portion of our data to the model training. In the above case, half the dataset does not contribute to the training of the model! This is not optimal, and can cause problems – especially if the initial set of training data is small. One way to address this is to use cross-validation; that is, to do a sequence of fits where each subset of the data is used both as a training set and as a validation set. Generally, predictions made on data not used to train a model provide insight into how the model will generalize to new situations. As such, scores that evaluate these predictions are referred to as the generalized performance of a machine learning model.
A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model. The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix. Accuracy is a metric used in classification problems used to tell the percentage of accurate predictions. We calculate it by dividing the number of correct predictions by the total number of predictions. The default form of accuracy gives an overall metric about model performance on the whole dataset. However, overall accuracy can be misleading when the class distribution is imbalanced, and it is critical to predict the minority class correctly.

IV. CONCLUSION

The classification template incorporated in this paper can be used for training and testing models having much larger datasets. This one template can be used for classification of different types of datasets where different ML algorithm’s accuracy can be tested out. The methods incorporated this project can be altered and developed in order to also solve problems of regression and clustering. It can form many ways for the efficient working of Artificial Intelligence. These contribute to creating correct algorithms and software, which enables the machine to act accordingly to the input program or data.

V. REFERENCES

[1]. Costa, E., Halpern, D., “The behavioural science of online harm and manipulation, and what to do about it,” The Behavioral Insights Team, 2019. Accessed May 27, 2020.
[2]. Coursera, Deep Learning Specialization. Accessed October 2020.
[3]. Davenport, T. H., Ronanki, R., “Artificial Intelligence for the Real World,” 2018. Accessed July 21, 2020.
[4]. Department of Defense, Directive 3000.09: Autonomy in Weapon Systems, November 21, 2012. DoD Directive 3000.09. Accessed May 27, 2020.
[5]. Domingos, P., The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books, 2015.
[6]. Etzioni, A., Etzioni, O. “Incorporating Ethics into Artificial Intelligence.” J Ethics 21, 403–418 (2017).
[7]. Ghorbani, Amirata, Abubakar, Abid, & Zou, James. “Interpretation of neural networks is fragile.” Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019.
[8]. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron, Deep Learning. MIT Press, 2016.
[9]. Hardesty, L., “Probabilistic programming does in 50 lines of code what used to take thousands.” Accessed July 21, 2020.
[10]. Hardt, Moritz, Price, Eric & Srebro, Nati. “Equality of opportunity in supervised learning.” Advances in neural information processing systems. 2016.
[11]. Hastie, T., Tibshirani, R., & Friedman, J., The Elements of Statistical Learning, Second Edition Springer; e-book. Accessed July 21, 2020.
[12]. Hayes, B., “Programming Languages Most Used and Recommended by Data Scientists.” Henke, N., et. al., “The age of analytics: Competing in a data-driven world,” McKinsey Global Institute, December 2016. Accessed October 2020.
[13]. Insight Data Science Fellows Program. Accessed October 2020
[14]. A Survey on Analysis of Data centre Performance and QOS in IaaS Cloud ,K Sharavana

[15]. Track-Before-Detect Technique Technique Technique for Mitigating Sea Mitigating Sea Clutter Based on Hough Transform ased on Hough Transform ased on Hough Transform M Chittapur, K Sharavana

[16]. SURVEY OF OPEN SOURCE IN THE CLOUD FOR FUTURE-THINKING U Muthaiah, K Sharavana, K Krishnamoorthi

[17]. Survey on Traffic Redundancy and Elimination Approach for Reducing Cloud Bandwidth and Costs S Chougala, K Sharavana

[18]. Improve Fault Tolerance and Dependable Data Integrity Protection of Storage System in Cloud of Clouds N Sharma, K Sharavana

[19]. A SURVEY ON DEPLOYING HPC IN THE CLOUD USING VOLUNTEER COMPUTING, International Journal of Computer Engineering and Applications, Sharavana .K L. Rahul

[20]. Deployment of Virtual HPC Clusters on Demand from Volunteer Computing Resources, International Journal of Advanced Research in Computer and Communication Engineering, Sharavana .K L. Rahul

[21]. Design of Traffic Redundancy and Elimination Approach for Reducing Cloud Bandwidth and Costs, international journal of innovative research in science and engineering, Suresh Chougala, K Sharavana

[22]. Track-before-detect technique for mitigating sea clutter based on hough transform, IJCSN International Journal of Computer Science and Network, M Chittapur, K Sharavana

[23]. Cold storage management system for farmers based on IoT, Int. J. Recent Trends Eng. Res, D Venkatesh, M Tatti, PG Hardikar, SS Ahmed, K Sharavana

[24]. Syed Mohammed Idris, Mohammed Kamran, Mohammed Saqeeb Khan M K, Arshadkhan A Jahagirda, Prof.Soumyashree Pattar, Prof.Megha S, "Debris Assortment Rover Using IOT & Video Processing Solid Dry Waste Collector", International Journal Of Modern Trends in Engineering and Research, Volume:9 Issue:3 Mar ' 2022.

[25]. Prof. Sharavana K, Prof. Asghar Pasha, Mohammed Saqeeb Khan M K, Syed Mohammed Idris, Mohammed Kamran, Arshadkhan A Jahagirdar, “Debris Assortment Rover Using IoT & Video Processing”, International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.9, Issue 6, page no.k241-k246, June-2022.