Iris Flower Species Classification

Student Name, 9867

Course name, Assignment …
VIT address
Supervisor/Lecturer name

Abstract

This report contains the summary and findings from the implementations of Supervised Machine Learning technique called Classification on the Iris data set [1] using Logistic Regression [2]. Different Exploratory Data Analysis Techniques (EDA) have been implemented on the data to get the in depth knowledge of data like distributions of columns, correlations among variables, hidden patterns, effect of variables on the resulting classes etc. Simple Logistic Regression model has been chosen for this type of multi-class classification (3 in this case) as the data does not possess any complexity. Iris flowers data has been collected from Kaggle [3] and the implementations are done using Python [4] and its libraries mainly pandas[5] for data manipulation, matplotlib [6] and seaborn [7] for visualizations, scipy [8] for statistical methods and scikit-learn [9] for different machine learning tasks. There are different steps discussed and implemented below in this process such as getting the input, data cleaning, EDA. model building and evaluation.

I. Introduction

MACHINE Learning has recently claimed its position in most of the industries worldwide due to the fact that people are producing more data everyday than ever before and very high computational rich systems. It can be used in any field to do any kind of work ranging from simple abbreviations recommendation to the very sophisticated self driving cars and human like robots. Supervised learning is a part of Machine Learning where intelligent models are build which solve the classification task by finding the patters such as classification of spam emails, image recognition, speech recognition etc. This report demonstrates the working of one such classification technique called Logistic Regression on the Iris flowers data.

Different statistical methods have been implemented and explained as well in the coming sections to demonstrate the need of EDA in Machine Learning as it is the part where we find the hidden patterns and dependencies among the whole data.

Our Iris data set is a data collected from 150 flowers where each flower belongs to one of the three classes Virginica, Setosa and Versicolor. There are 4 columns/features/attributes which define the Petal Length, Petal Width, Sepal Length, Sepal Width of every flower. We have to build a machine learning model that can correctly classify the class of a new flower given these attributes. Analysis and findings on the data is discussed in the section below.

II. Discussion and Analytic

Our data has 4 attributes columns and 1 label column. After the first EDA, there were no missing values found in any sense so we can say that the data is free from missing values. Figure 1 shows that there some outliers present in SepalWidth column. Average value of SepalLength is relatively higher than the other features.

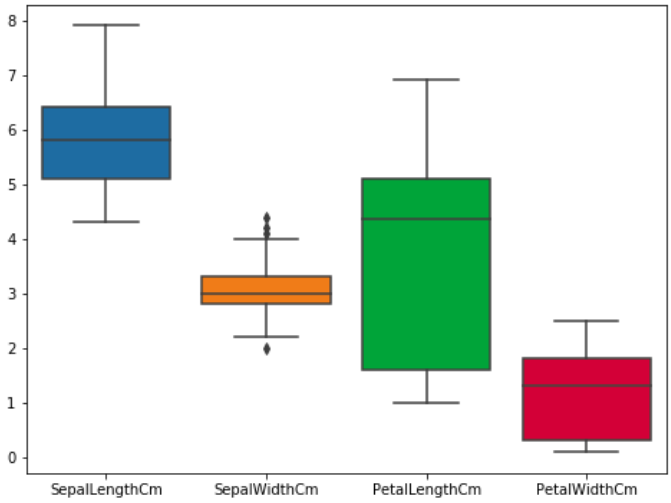

Figure 1: Box Plots of Features

Distributions of the classes in the data in perfect as every class has 33.33% of the data and can be seen in Figure 2 so we don not have to worry about the over and under sampling problem about the data that could produce a bias in our model. Pairplot depicted in Figure 3 describes the class vise distribution and relation of each and every numerical variable to every other based on the classes. It is evident that SepalLength and PetalLength are highly correlated as there seems to be a positive correlation between the two. Same can be seen between PetalLength and PetalWidth as there is a clear upward linear positive pattern depicted. From this figure, we can even see the distributions of the attributes related to different classes. Most of the flowers from Setosa class have a very low PetalLength and PetalWidth as there is a peak in the

distribution. On the other hand, Virginica and Versicolor domain is wider as the flowers show more variances as the class distributions are wide.

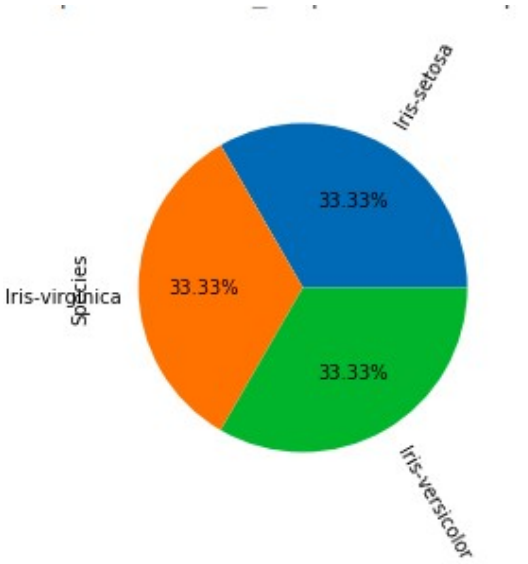

Figure 2: Pie Plot of the classes

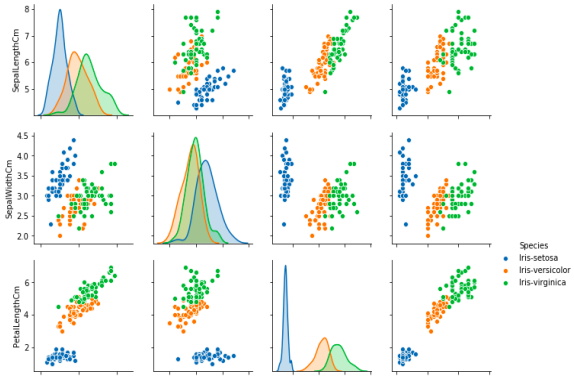

Figure 3: Pairplots of attributes

It seems like there are more than one correlated features and it is never a good thing to have in the model so we can confirm it by looking at the Heatmap in Figure-4 of the correlation matrix that gives us the correlation of each column to every other column. This figure tells us that PetalWidth and PetalLength are highly correlated with a score of 0.96 and same holds for the PetalLength ans SepalLength wit h 0.87. Apart from that, the figure proves our hypothesis that PetalLength and PetalWidth are important in decision of the label because they have correlation value of 0.96 and 0.95 respectively.

We can confirm our hypothesis even more substantially by looking at the Radial and Parallel Coordinates plots which will tell us that which attributes are more important than the others in the prediction of classes.

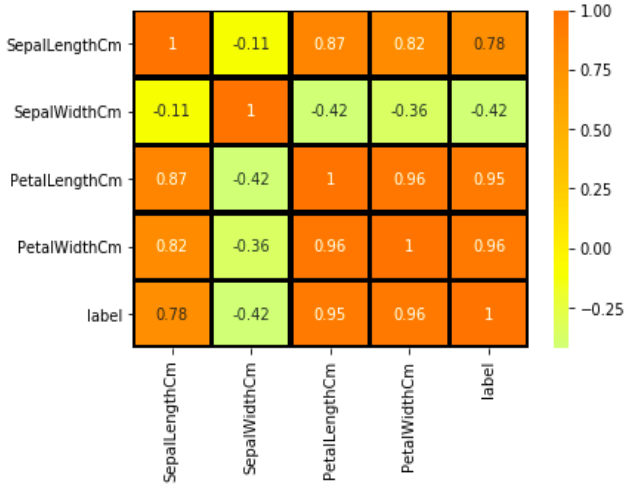

Figure 4: Heatmap of the Correlation Matrix

We can clearly see in Figure-5 that Virginica class tend to have higher PetalLength, higher SepalLength but lower SeplLength on average. On the other hand, Setosa has the lowest PetalLength and PetlWidth. So we can see that we can define the class Setosa by only these two variables as this class is clearly separated from others with no overlapping between points.

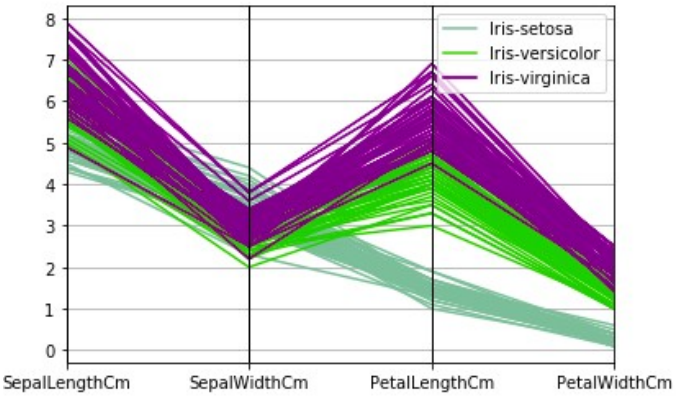

Figure 5: Parallel Coordinates

There are some abnormalities in our data in terms of normal distribution and we can visualize that PetalLength and PetalWidth are bimodal in nature in Figure-6. Proof of the same can be observed in Figure 7 where a normal probability plot shows deviation from the normal line. SepalWidth is almost perfectly normal with a single peak in the middle with little to no skewness. On the other hand, SepalLength looks confusing because there seems to be two different peaks but as they are close enough to each other, it gives us a flat probability density function and gives the effect of flatness in the distribution. In he coming section, we will apply Logistic Regression by using the methods in scikit-learn to predict from the cross validation and test test separated beforehand. We will be evaluating our model on the basis of accuracy and other metrics already provided by the scikit-learn library.

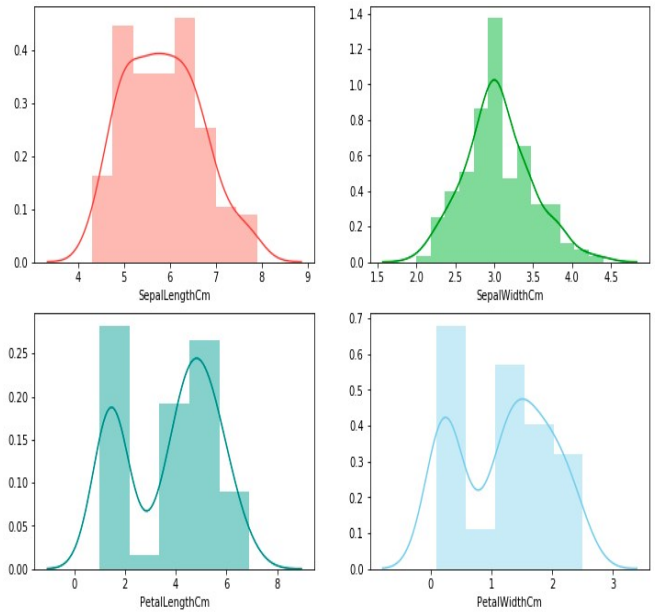

Figure 6: Histograms with Probability Density Functions

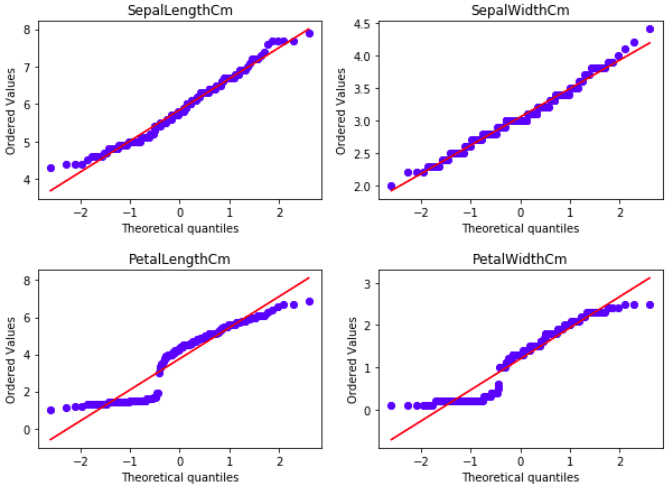

Figure 7: Normality Plots

III. CONCLUSION

After implementing Logistic Regression with given default parameters, we get an accuracy score of 0.93 on the training data and 0.95 on the test data. Higher accuracy score in testing than the training shows that our model is not over fitting. With a random data set with all the four features given, we can say that our model will predict the class correctly 95 times out of 100 attempts. That is fairly a god result given Logistic Regression is inarguably the simplest classification approach. The model provides us with the coef_ values for the attributes which show us the importance of each variable. This proves our hypothesis about PetalWidth and PetalLength as they have the highest values among all the coef_. We can try to drop one of these features as PetalLength is highly correlated with PetalWidth as it will only inflate the variance with little or no contribution on the results.

REFERENCES

[1] https://archive.ics.uci.edu/ml/datasets/iris
[2] RPeng, C.-Y. J, Lee, K Land G.M. Ingersoll, GM (2002) “An introduction to logistic regression analysis and reporting,” The Journal of Educational Research”,96(1),3-14. http://bit.csc.lsu.edu/~jianhua/emrah.pdf.
[3] https://www.kaggle.com/uciml/iris
[4] https://www.python.org/
[5] https://pandas.pydata.org/
[6] https://matplotlib.org/
[7] https://seaborn.pydata.org/
[8] https://www.scipy.org/
[9] https://scikit-learn.org/stable/

APPENDIX: PYTHON IMPLEMENTATION

Below is the code for implementation in Python. Indentation should be there to completely run the code.

```
# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import norm
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import
accuracy_score,r2_score,recall_score,precision_score

import warnings # current version of seaborn generates a
bunch of warnings that we'll ignore
warnings.filterwarnings("ignore")
SEED = 13
np.random.seed(SEED) # set a stable state to produce
reproducible code


iris = pd.read_csv('Iris.csv')
iris.info()
iris.head()
iris.describe()
iris.isnull().sum()
iris.drop('Id',inplace=True,axis=1) # drop the column 'Id' as it
has no use
iris["Species"].value_counts()
```

```
iris["Species"].value_counts().plot(kind='pie',autopct='%1.2f
%%',rotatelabels=True)

sns.pairplot(iris, hue="Species", size=2.3)

class_values = iris['Species'].value_counts().index.tolist() #
unique target classes
f,ax = plt.subplots(2,2,figsize=(12,8))
ax = ax.ravel()
for i,column in
enumerate(iris.drop('Species',axis=1).columns.tolist()):
    for j in class_values:
        ax[i].set_xlabel(column)
        sns.kdeplot(iris.loc[iris['Species'] == j, column], label =
j,ax=ax[i],shade=True)

fig = plt.figure(figsize=(8,6))
sns.boxplot(data=iris)

c_cmap = ['tomato','green','teal','skyblue']
f,ax = plt.subplots(2,2,figsize=(12,8))
ax = ax.ravel()
for i,column in
enumerate(iris.drop('Species',axis=1).columns.tolist()):
    ax[i].set_xlabel(column)
    sns.distplot(iris[column],ax=ax[i],color=c_cmap[i],)

c_cmap = ['tomato','green','teal','skyblue']
f,ax = plt.subplots(2,2,figsize=(10,7))
ax = ax.ravel()
plt.subplots_adjust(bottom=None, right=None, top=None,
wspace=None, hspace=0.4)
for i,column in
enumerate(iris.drop('Species',axis=1).columns.tolist()):
    stats.probplot(iris[column], plot=ax[i])
    ax[i].set_title(column)


f,ax = plt.subplots(2,2,figsize=(12,8))
ax = ax.ravel()
for i,column in
enumerate(iris.drop('Species',axis=1).columns.tolist()):
    for j in class_values:
        ax[i].set_xlabel(column)
        sns.violinplot(x="Species", y=column,
data=iris,inner=None,color='0.8',linewidth=3,ax=ax[i])
        sns.stripplot(x="Species", y=column, data=iris,
jitter=True,alpha=0.8,ax=ax[i])

f,ax = plt.subplots(2,2,figsize=(12,8))
ax = ax.ravel()
for i,column in
enumerate(iris.drop('Species',axis=1).columns.tolist()):
    for j in class_values:
        ax[i].set_xlabel(column)
        sns.boxplot(y=column,x='Species',data=iris,ax=ax[i])

from pandas.plotting import parallel_coordinates
parallel_coordinates(iris, "Species")

iris['label'] = LabelEncoder().fit_transform(iris['Species'])
iris.corr()
sns.heatmap(iris.corr(),cmap="Wistia",lw=2.2,linecolor='blac
k',annot=True)
b, t = plt.ylim() # discover the values for bottom and top
b += 0.5 # Add 0.5 to the bottom
t -= 0.5 # Subtract 0.5 from the top
plt.ylim(b, t) # update the ylim(bottom, top) values
plt.show()

X = iris.drop(['Species','label'],axis=1)
y = iris['label']
std_x = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test =
train_test_split(std_x,y,test_size=0.2,random_state=SEED)

clf =
LogisticRegression(multi_class='auto',random_state=SEED).fi
t(X_train,y_train)
clf.score(X_train,y_train)
accuracy_score(y_test,clf.predict(X_test))
```