IE406 Machine Learning

Lab Assignment - 3

Group 39

201901201: Sharvil Sheth
201901216: Prabhav Shah
201901301: Mohil Desai
201901462: Dhaval Vaidya
201901464: Vishvesh Patel

Question 1

In this question we will use the data given in file "Social_Network_Ads.csv" which is a categorical dataset to determine whether a user purchased a product or not by using three features to determine user's decision. Visualize the data by 3D plotting features using different colors for label 0 and 1. Use data in files "Social_Network_Ads.csv" to perform logistic regression by implementing logistic function and with available library function and compare your results. Use 90% data points from each set for training and remaining 10% for testing the accuracy of classification. Using confusion matrix find accuracy, precision, F1 score and recall.

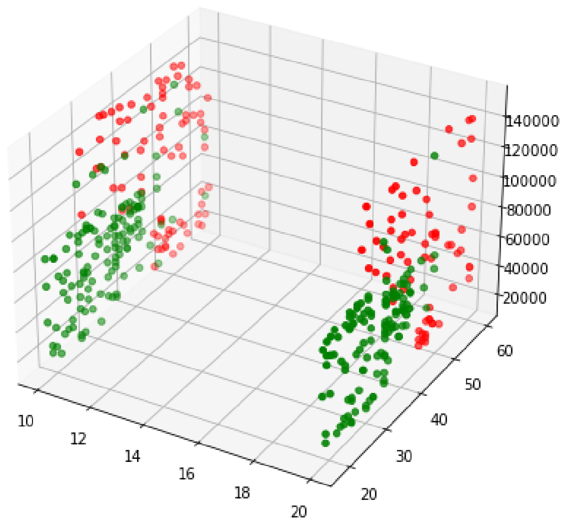
Answer

code

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
import scipy
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

df = pd.read_csv('Social_Network_Ads.csv')
df['Gender'] = df['Gender'].replace("Male", 20)
df['Gender'] = df['Gender'].replace("Female", 10)
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    int64
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(5)
memory usage: 15.8 KB

grouped = df.groupby(df.Purchased)
bias_0_plot = grouped.get_group(0)
bias_1_plot = grouped.get_group(1)

X_0_plot = bias_0_plot[['Gender', 'Age', 'EstimatedSalary']].values

X_1_plot = bias_1_plot[['Gender', 'Age', 'EstimatedSalary']].values

X_0_norm0 = preprocessing.normalize([X_0_plot[:,0]])
X_0_norm1 = preprocessing.normalize([X_0_plot[:,1]])
X_0_norm2 = preprocessing.normalize([X_0_plot[:,2]])
```

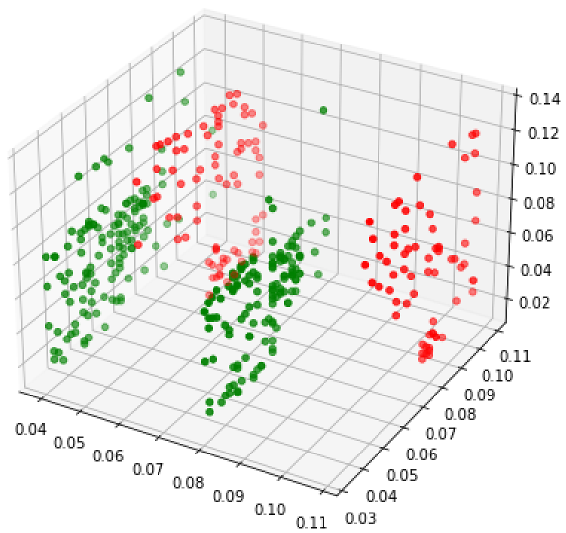
```
X_1_norm0 = preprocessing.normalize([X_1_plot[:,0]])
X_1_norm1 = preprocessing.normalize([X_1_plot[:,1]])
X_1_norm2 = preprocessing.normalize([X_1_plot[:,2]])

fig = plt.figure(figsize = (10, 7))
ax = plt.axes(projection ="3d")

ax.scatter3D(X_0_plot[:,0], X_0_plot[:,1], X_0_plot[:,2], color = "green")
ax.scatter3D(X_1_plot[:,0], X_1_plot[:,1], X_1_plot[:,2], color = "red")
plt.show()
```

Listing 1: Question 1




```
fig = plt.figure(figsize = (10, 7))
ax = plt.axes(projection ="3d")

ax.scatter3D(X_0_norm0, X_0_norm1, X_0_norm2, color = "green")
ax.scatter3D(X_1_norm0, X_1_norm1, X_1_norm2, color = "red")
plt.show()
```

Listing 2: Question 1




```

def sigmoid(x):
    # Activation function used to map any real value between 0 and 1
    return 1 / (1 + np.exp(-x))

```

```
def net_input(theta, x):
    # Computes the weighted sum of inputs
    return np.dot(x, theta)

def probability(theta, x):
    # Returns the probability after passing through sigmoid
    return sigmoid(net_input(theta, x))

    def cost_function(theta, x, y):
    # Computes the cost function for all the training samples
    m = x.shape[0]
    total_cost = -(1 / m) * np.sum(y * np.log(probability(theta, x)) + (1 - y) * np.log(1 -
    probability(theta, x)))
    return total_cost

def gradient(theta, x, y):
    # Computes the gradient of the cost function at the point theta
    m = x.shape[0]
    return (1 / m) * np.dot(x.T, sigmoid(net_input(theta,   x)) - y)

    def fit(x, y, theta):
    opt_weights = scipy.optimize.fmin_tnc(func=cost_function, x0=theta,
                  fprime=gradient,args=(x, y.flatten()))
    return opt_weights[0]

    X = df.iloc[:, 1:4]
X = np.c_[np.ones((X.shape[0], 1)), X]
Y = df[['Purchased']].values
theta = np.zeros((X.shape[1], 1))

X_train_own, X_test_own, Y_train_own, Y_test_own = train_test_split(X, Y, test_size=0.2)

parameters = fit(X_train_own, Y_train_own, theta)
parameters
output - array([-1.54919397e+01,  2.61066273e-02,  2.97935984e-01,  3.83055362e-05])

def predict(x):
    theta = parameters[:, np.newaxis]
    return probability(theta, x)

def accuracy(x, actual_classes, probab_threshold=0.5):
    predicted_classes = (predict(x) >=
                         probab_threshold).astype(int)
    predicted_classes = predicted_classes.flatten()
    accuracy = np.mean(predicted_classes == actual_classes)
    return accuracy

accuracy(X_test_own, Y_test_own.flatten())
output- 0.7875
```

Listing 3: Logistic Regression our own implementation

```
X = df[['Gender','Age',"EstimatedSalary"]].values
Y = df['Purchased'].values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

clf = LogisticRegression(random_state=0,max_iter=1000).fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
clf.score(X_test,Y_test)
output- 0.65

tn, fp, fn, tp = confusion_matrix(Y_test, Y_pred).ravel()

acc = (tn+tp)/(tn+tp+fp+fn)
prec = tp/(tp+fp)
recall = tp/(tp+fn)
f1_score = 2*(prec*recall)/(prec+recall)
print('Accuracy - ',acc)
print('Precision - ',prec)
print('Recall - ',recall)
print('F1 Score - ',f1_score)

Accuracy -  0.65
Precision -  0.5151515151515151
```

```
Recall -  0.5862068965517241
F1 Score -  0.5483870967741935
```

Listing 4: Implementing sklearn logistic regression

Observation/ Justification

In this question we observed that the logistic regression implementation that we did on our own gave better performance than the sklearn library

Question 2

You will work with a widely used Iris dataset. The Iris Dataset contains four features (sepal length, sepal width, petal length, and petal width) of 50 samples of three species of Iris (Iris setosa, Iris virginica, and Iris versicolor). Plot features' histogram. Compute pdf and compare it with histogram. perform the exploratory data analysis by plotting the basic statistics like mean, median, min, and max value of each feature (sepal and petal lengths and widths) for each of the three classes (setosa, virginica, and versicolor).

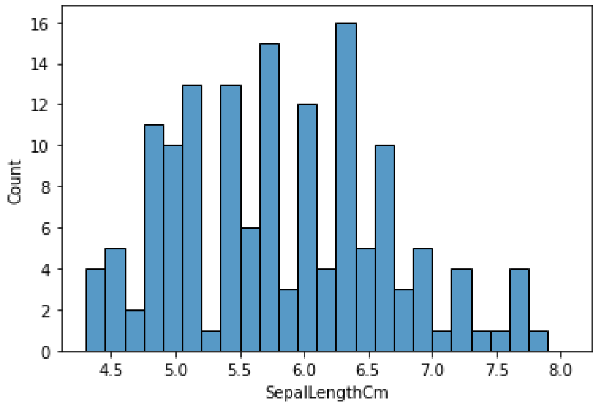
Answer

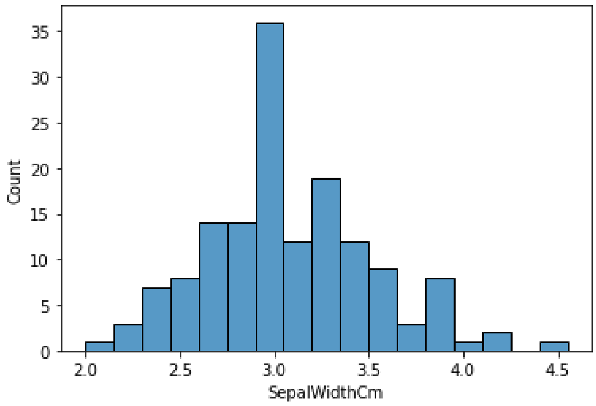
code

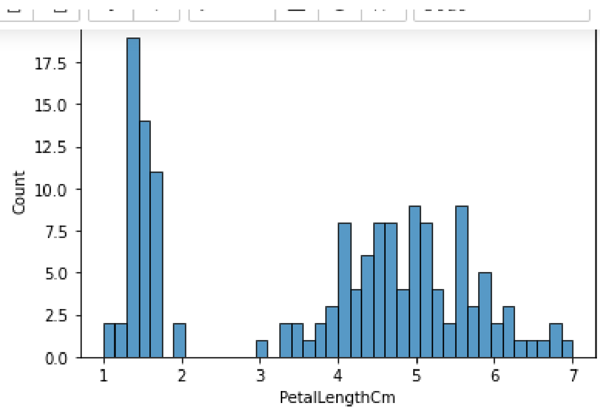
```python
#Code for plotting Histograms
import seaborn as sns
plt . figure ()
sns . histplot ( data =df[df. columns [1]] , binwidth =0.15)
plt . figure ()
sns . histplot ( data =df[df. columns [2]] , binwidth =0.15)
plt . figure ()
sns . histplot ( data =df[df. columns [3]] , binwidth =0.15)
plt . figure ()
sns . histplot ( data =df[df. columns [4]] , binwidth =0.15) ;
```

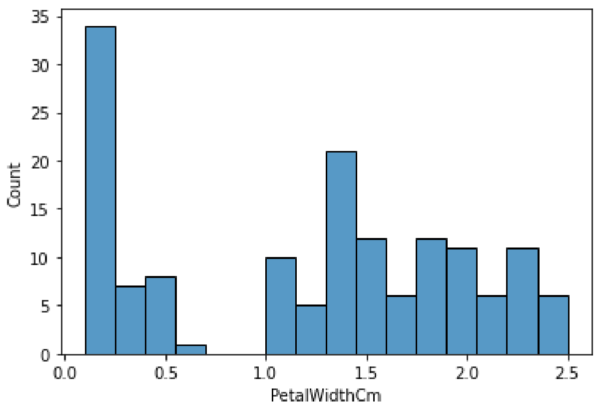
Listing 5: Question 2

Result









code

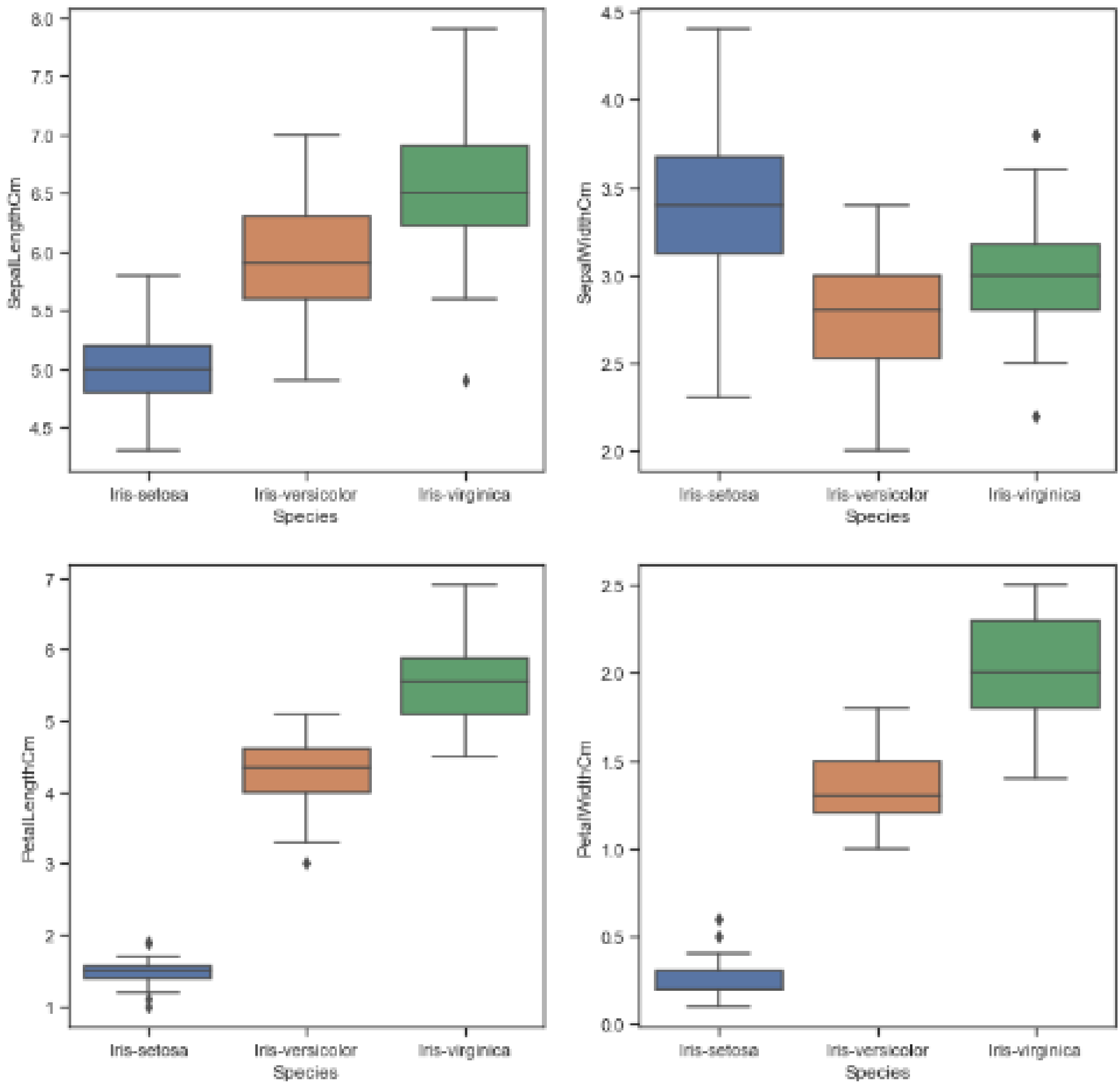
```python
#Code for basic statistics
iris=df.iloc[:,1:]
iris.groupby('Species').agg(['mean','median','min','max'])

#Code for boxplot to plot and analyse the basic stats
import seaborn as sns
iris=df.iloc[:,1:]
sns.set(style="ticks")
plt.figure(figsize=(12,12))
plt.subplot(2,2,1)
sns.boxplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.boxplot(x='Species',y='SepalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.boxplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.boxplot(x='Species',y='PetalWidthCm',data=iris)
plt.show()
```

Listing 6: Question 2

Result

	SepalLengthCm				SepalWidthCm				PetalLengthCm				PetalWidthCm			
	mean	median	min	max	mean	median	min	max	mean	median	min	max	mean	median	min	max
Species																
Iris-setosa	5.006	5.0	4.3	5.8	3.418	3.4	2.3	4.4	1.464	1.50	1.0	1.9	0.244	0.2	0.1	0.6
Iris-versicolor	5.936	5.9	4.9	7.0	2.770	2.8	2.0	3.4	4.260	4.35	3.0	5.1	1.326	1.3	1.0	1.8
Iris-virginica	6.588	6.5	4.9	7.9	2.974	3.0	2.2	3.8	5.552	5.55	4.5	6.9	2.026	2.0	1.4	2.5

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Species				
Iris-setosa	0.352490	0.381024	0.173511	0.107210
Iris-versicolor	0.516171	0.313798	0.469911	0.197753
Iris-virginica	0.635880	0.322497	0.551895	0.274650



Question 3

Visualize the data in the Iris Dataset by considering maximum combinations of two features in a 2D plot. Use red, green, and blue colors for labeling the three classes: Iris setosa, Iris virginica, and Iris versicolor, respectively. Comment on whether any two classes among the three can be separated by a line? Report your observations for each case.

Answer

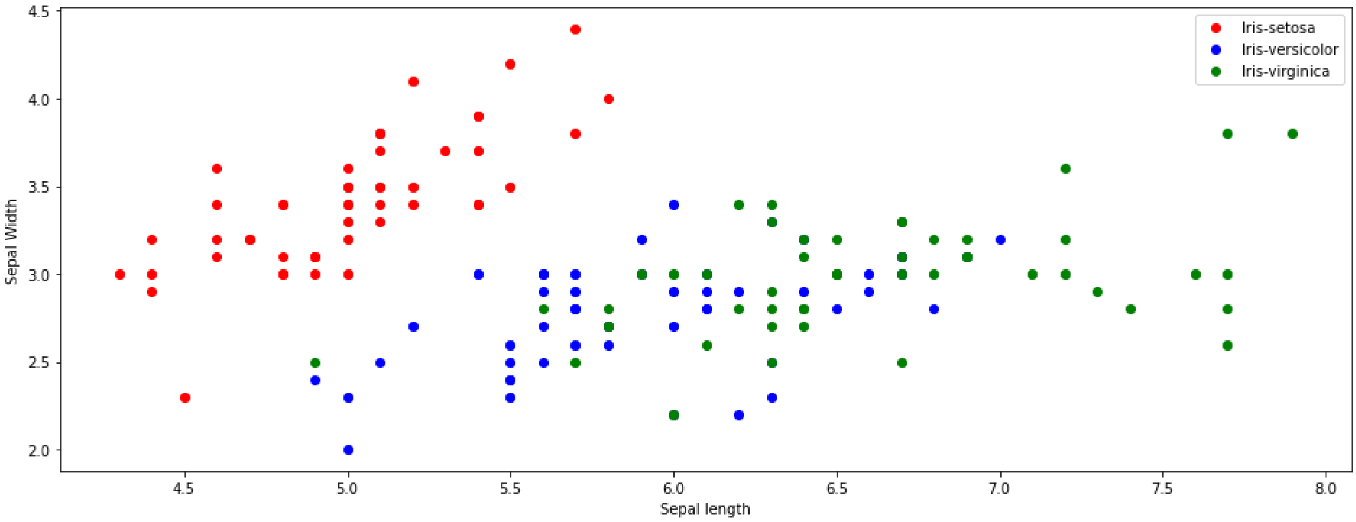
code

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.linalg as la
import math
from scipy.fftpack import fftfreq
from scipy.linalg import toeplitz
from matplotlib import animation
```

```python

df = pd.read_csv(r'C:\Users\Vatsal\Downloads\Iris.csv')
X=df.iloc[:,1:3]
Y=df.iloc[:,5]
X=np.array(X)
Y=np.array(Y)
a=X[np.where(Y=='Iris-setosa')]
b=X[np.where(Y=='Iris-versicolor')]
c=X[np.where(Y=='Iris-virginica')]
plt.figure(figsize=(16,20))
plt.subplot(3, 1, 1)
plt.scatter(a[:,0],a[:,1],c='red',alpha=1,label='Iris-setosa')
plt.scatter(b[:,0],b[:,1],c='blue',alpha=1,label='Iris-versicolor')
plt.scatter(c[:,0],c[:,1],c='green',alpha=1,label='Iris-virginica')
plt.legend()
plt.xlabel('Sepal length')
plt.ylabel('Sepal Width')
plt.show()
```

Listing 7: Question 3

Result



Observation

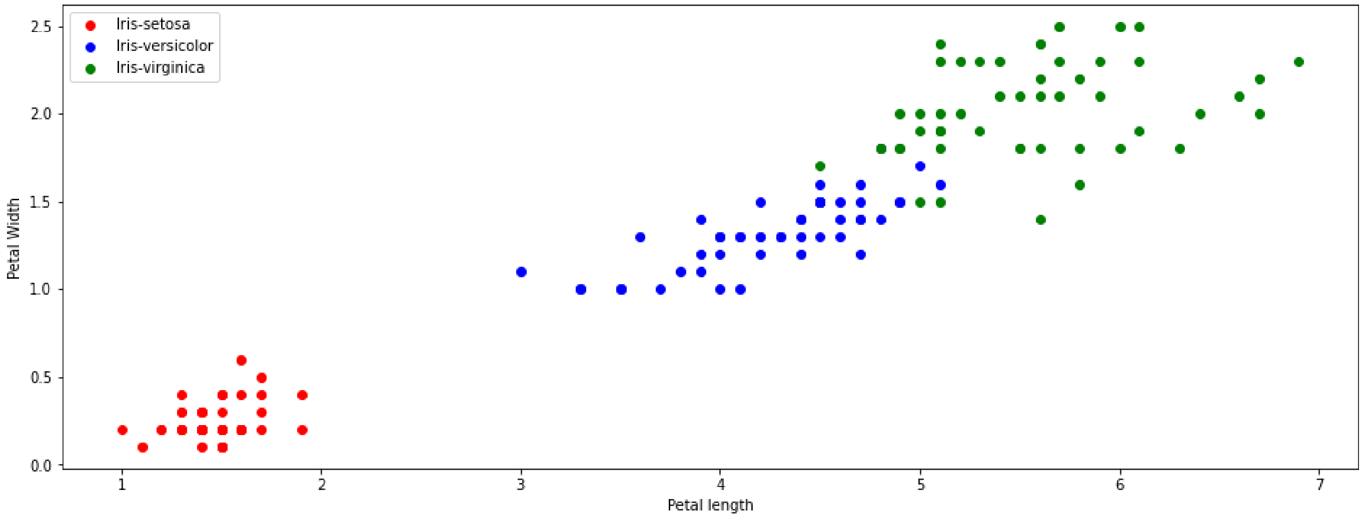
Class setosa and Class versicolor can be separated by drawing a line and we can get 100% training accuracy as both class dots are not overlapping with eachother in majority of area. Similarly, class setosa and class virginica can be separated by drawing a line and can achieve a 100% train accuracy. Here, class versicolor and class virginica are overlapping and cannot be separated by a line.

code

```python
X=df.iloc[:,3:5]
Y=df.iloc[:,5]
X=np.array(X)
Y=np.array(Y)
a=X[np.where(Y=='Iris-setosa')]
b=X[np.where(Y=='Iris-versicolor')]
c=X[np.where(Y=='Iris-virginica')]
plt.figure(figsize=(16,20))
plt.subplot(3, 1, 1)
plt.scatter(a[:,0],a[:,1],c='red',alpha=1,label='Iris-setosa')
plt.scatter(b[:,0],b[:,1],c='blue',alpha=1,label='Iris-versicolor')
plt.scatter(c[:,0],c[:,1],c='green',alpha=1,label='Iris-virginica')
plt.legend()
plt.xlabel('Petal length')
plt.ylabel('Petal Width')
plt.show()
```

Listing 8: Question 3

Result



Observation/ Justification

Class setosa and Class versicolor can be separated by drawing a line and we can achieve 100% train accuracy as both class dots are not overlapping with majority area of eachother. Similarly, Class setosa and Class virginica can be separated by drawing a line and can achieve a 100% train accuracy.Here ,Class versicolor and Class Virginica are separable with a line, but cant guarantee 100% train accuracy since there are some dots overlapping. Hence this dots near decision boundary will not guarantee completely accurate predictions.

Question 4

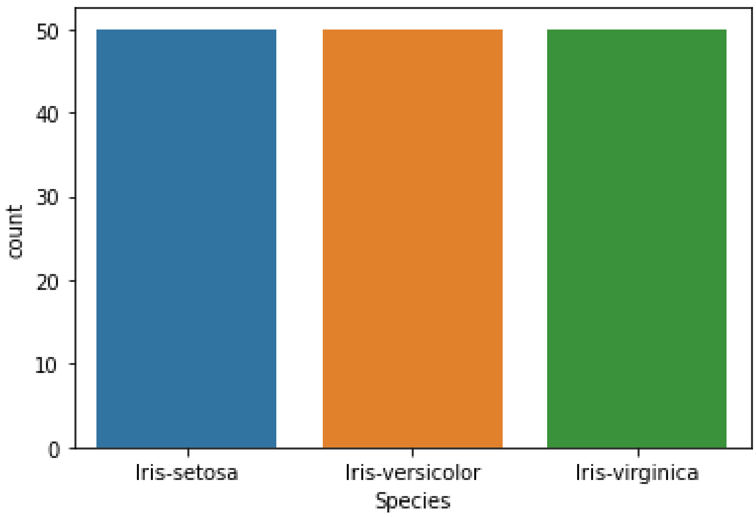
Perform logistic regression on IRIS Dataset and plot confusion matrix. Using confusion matrix find accuracy, precision, F1 score and recall.

Answer

code

```

import matplotlib.pyplot as p
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score
from sklearn.metrics import confusion_matrix

data = pd.read_csv('Iris.csv')
p.figure(1)
sns.countplot(x = 'Species', data = data)

data.drop('Id', axis=1, inplace=True)
```

Listing 9: Question 4




```
# Changing
# Iris-setosa to 0
# Iris-versicolor to 1
# Iris-virginica to 2


for i in range (len(data['Species'])):
    if (data['Species'][i] == 'Iris-setosa'):
        data['Species'][i] = 0
    elif (data['Species'][i] == 'Iris-versicolor'):
        data['Species'][i] = 1
    elif (data['Species'][i] == 'Iris-virginica'):
        data['Species'][i] = 2
```

Listing 10: Question 4

```

X = data.drop(['Species'], axis = 1)
y = data['Species']
y = y.astype('int')

##  CHANGE TEST SIZE HERE
testSize = 0.5


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=testSize, random_state= 40
    )
```

Listing 11: Data Splitting to Test and Training

```

logmodel = LogisticRegression(max_iter=1000)
logmodel.fit(X_train, y_train)

predictions = logmodel.predict(X_test)
```

Listing 12: Training Data and Predictions

```

report = classification_report(y_test, predictions)
print(report)

acc = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='micro')
recall = recall_score(y_test, predictions, average='micro')
f1 = f1_score(y_test, predictions, average='micro')
print("Accuracy: " ,acc)
print("Precision: ", precision)
print("Recall: ", recall)
print("f1 score: ", f1)


conf_matrix = confusion_matrix(y_test, predictions)
```

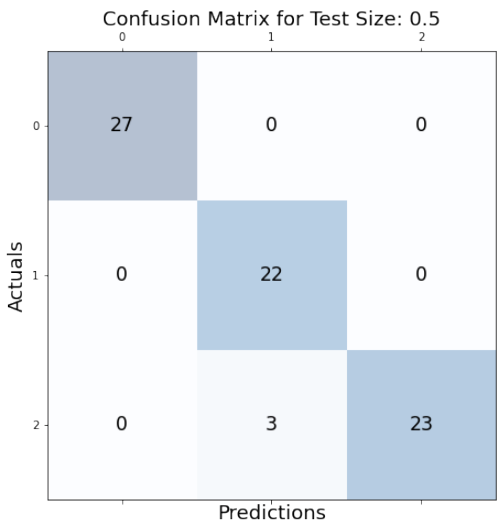
```
print('\n\nConfusion Matrix :-\n', confusion_matrix(y_test, predictions))

 precision    recall  f1-score   support

           0       1.00      1.00      1.00        27
           1       0.88      1.00      0.94        22
           2       1.00      0.88      0.94        26

    accuracy                           0.96        75
   macro avg       0.96      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75

Accuracy:  0.96
Precision:  0.96
Recall:  0.96
f1 score:  0.96


Confusion Matrix :-
 [[27  0  0]
 [ 0 22  0]
 [ 0  3 23]]

```

Listing 13: Confusion Matrix

```


 p.figure(3)

fig, ax = p.subplots(figsize=(7.5, 7.5))
ax.matshow(conf_matrix, cmap=p.cm.Blues, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')


p.xlabel('Predictions', fontsize=18)
p.ylabel('Actuals', fontsize=18)
p.title('Confusion Matrix for Test Size: ' + str(testSize) , fontsize=18)


p.show()
```

Listing 14: Confusion Matrix

Result



Observation/ Justification

Question 5

In this question you will perform logistic regression for multiclass classification on the 20 News groups dataset. Since this dataset is a balanced one, you will perform the pre- processing to create an imbalanced version of the dataset (by removing some news articles from some groups). One example is given below. Perform multiclass classification using logistic regression on both the balanced and the imbalanced version of the dataset. Compare the performance in each case by obtaining the confusion matrix and accuracy. Report you observations at the end.

Answer

code

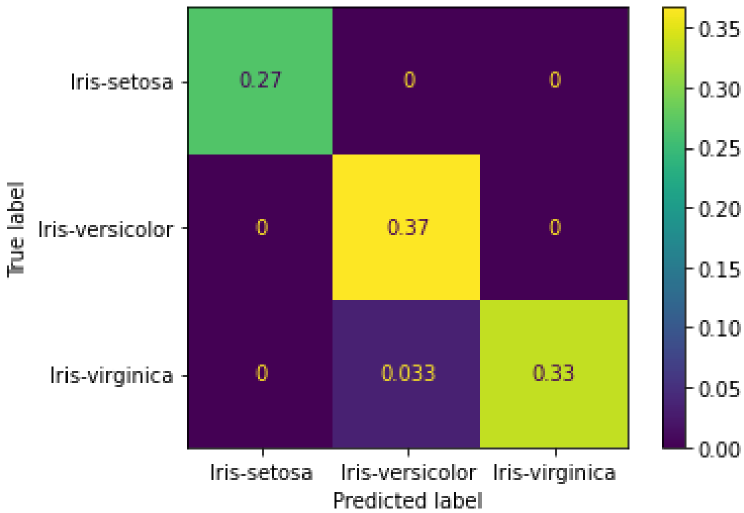
```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score

df = pd.read_csv('Iris.csv')
# df['Species'] = df['Species'].replace('Iris-setosa', 0)
# df['Species'] = df['Species'].replace('Iris-versicolor', 1)
# df['Species'] = df['Species'].replace('Iris-virginica', 2)
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Listing 15: Question 5

```
#loading data on X and Y
X = df.iloc[:, 1:5]
X = np.c_[np.ones((X.shape[0], 1)), X]
Y = df[['Species']].values
theta = np.zeros((X.shape[1], 1))

np.array(np.unique(Y, return_counts=True)).T

array([['Iris-setosa', 50],
       ['Iris-versicolor', 50],
       ['Iris-virginica', 50]], dtype=object)

       #spliting dataset for training and testing
X_train_bal, X_test_bal, Y_train_bal, Y_test_bal = train_test_split(X, Y, test_size=0.2)

#training model
clf_bal = LogisticRegression(random_state=0, max_iter=1000).fit(X_train_bal, Y_train_bal)
Y_pred_bal = clf_bal.predict(X_test_bal)

acc_bal = accuracy_score(Y_test_bal, Y_pred_bal)
precision_bal = precision_score(Y_test_bal, Y_pred_bal, average='micro')
recall_bal = recall_score(Y_test_bal, Y_pred_bal, average='micro')
f1_bal = f1_score(Y_test_bal, Y_pred_bal, average='micro')
print("Accuracy: " ,acc_bal)
print("Precision: ", precision_bal)
print("Recall: ", recall_bal)
print("f1 score: ", f1_bal)
```

```

report_bal = classification_report(Y_test_bal,Y_pred_bal)
print(report_bal)

Accuracy:  0.9666666666666667
Precision:  0.9666666666666667
Recall:  0.9666666666666667
f1 score:  0.9666666666666667
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       0.92      1.00      0.96        11
 Iris-virginica       1.00      0.91      0.95        11

       accuracy                           0.97        30
      macro avg       0.97      0.97      0.97        30
   weighted avg       0.97      0.97      0.97        30

   #generating confusion matrix
cm_bal = confusion_matrix(Y_test_bal, Y_pred_bal, labels=clf_bal.classes_, normalize='all')
disp_bal = ConfusionMatrixDisplay(confusion_matrix=cm_bal, display_labels=clf_bal.classes_)
disp_bal.plot()
```

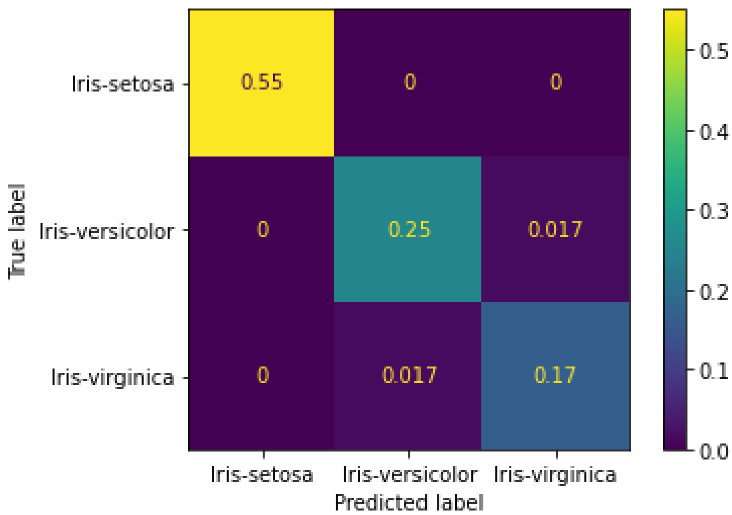
Listing 16: Multicalss classification on Balanced dataset




```
#making imbalanced dataset from balanced one
df_imbalance = df.copy()
df_imbalance = df_imbalance.append(df[0:50])
df_imbalance = df_imbalance.append(df[0:50])
df_imbalance = df_imbalance.append(df[50:100])

#loading data on X and Y
X = df_imbalance.iloc[:, 1:5]
X = np.c_[np.ones((X.shape[0], 1)), X]
Y = df_imbalance[['Species']].values
theta = np.zeros((X.shape[1], 1))

np.array(np.unique(Y, return_counts=True)).T

array([['Iris-setosa', 150],
       ['Iris-versicolor', 100],
       ['Iris-virginica', 50]], dtype=object)

       #spliting dataset for training and testing
X_train_imbal, X_test_imbal, Y_train_imbal, Y_test_imbal = train_test_split(X, Y, test_size=0.2)

#training model and calculating accuracy
clf_imbal = LogisticRegression(random_state=0, max_iter=1000).fit(X_train_imbal, Y_train_imbal)
Y_pred_imbal = clf_imbal.predict(X_test_imbal)

acc_imbal = accuracy_score(Y_test_imbal, Y_pred_imbal)
precision_imbal = precision_score(Y_test_imbal, Y_pred_imbal, average='micro')
```

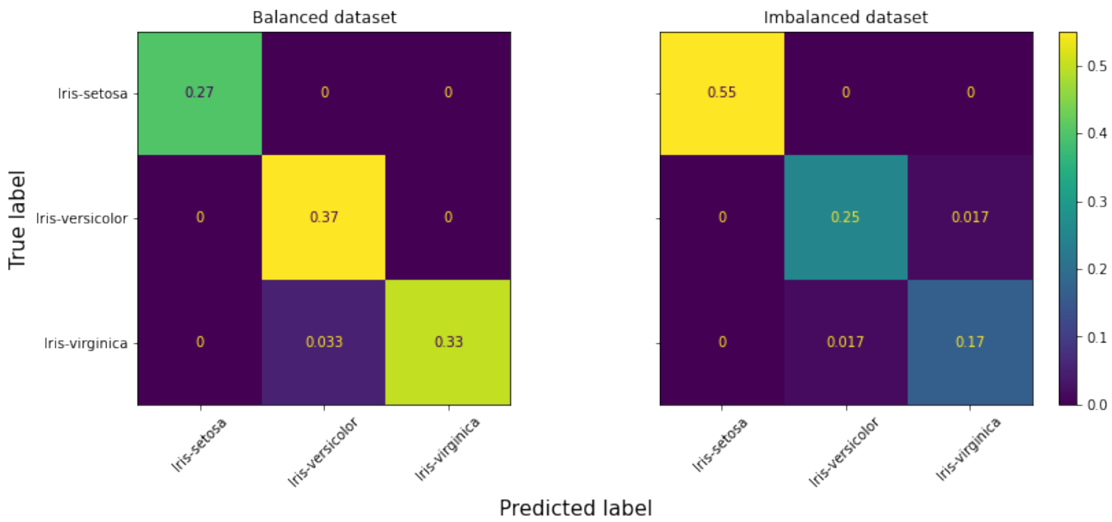
```
recall_imbal = recall_score(Y_test_imbal, Y_pred_imbal, average='micro')
f1_imbal = f1_score(Y_test_imbal, Y_pred_imbal, average='micro')
print("Accuracy: " ,acc_imbal)
print("Precision: ", precision_imbal)
print("Recall: ", recall_imbal)
print("f1 score: ", f1_imbal)

report_imbal = classification_report(Y_test_imbal,Y_pred_imbal)
print(report_imbal)

Accuracy:  0.9666666666666667
Precision:  0.9666666666666667
Recall:  0.9666666666666667
f1 score:  0.9666666666666667
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        33
Iris-versicolor       0.94      0.94      0.94        16
 Iris-virginica       0.91      0.91      0.91        11

       accuracy                           0.97        60
      macro avg       0.95      0.95      0.95        60
   weighted avg       0.97      0.97      0.97        60

   #generating confusion matrix
cm_imbal = confusion_matrix(Y_test_imbal, Y_pred_imbal, labels=clf_imbal.classes_, normalize='all')
disp_imbal = ConfusionMatrixDisplay(confusion_matrix=cm_imbal, display_labels=clf_imbal.classes_)
disp_imbal.plot()
```

Listing 17: Multiclass classification on imbalanced data set




```
f, axes = plt.subplots(1, 2, figsize=(15,5), sharey=True)

disp_bal.plot(ax=axes[0], xticks_rotation=45)
disp_bal.ax_.set_title("Balanced dataset")
disp_bal.im_.colorbar.remove()
disp_bal.ax_.set_ylabel('True label', fontsize=15)
disp_bal.ax_.set_xlabel('')

disp_imbal.plot(ax=axes[1], xticks_rotation=45)
disp_imbal.ax_.set_title("Imbalanced dataset")
disp_imbal.ax_.set_ylabel('')
disp_imbal.ax_.set_xlabel('')

f.text(0.47, -0.1 ,'Predicted label', fontsize=15)
plt.show()
```

Listing 18: Observation



Balanced dataset
Imbalanced dataset
True label
Iris-setosa
Iris-versicolor
Iris-virginica
0.27
0
0
0
0.37
0
0
0.033
0.33
0.55
0
0
0
0.25
0.017
0
0.017
0.17
0.5
0.4
0.3
0.2
0.1
0.0
Iris-setosa
Iris-versicolor
Iris-virginica
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted label