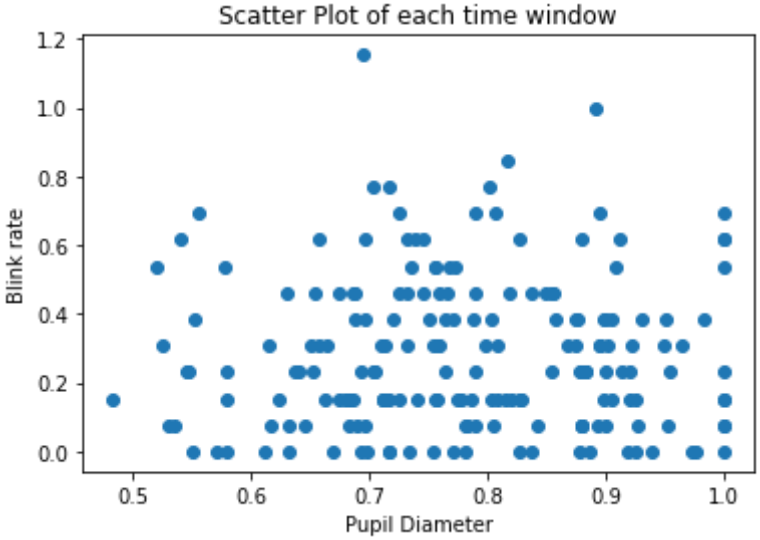
In [4]:
```
#Installing Scikit Learn
!pip install --user scikit-learn
```

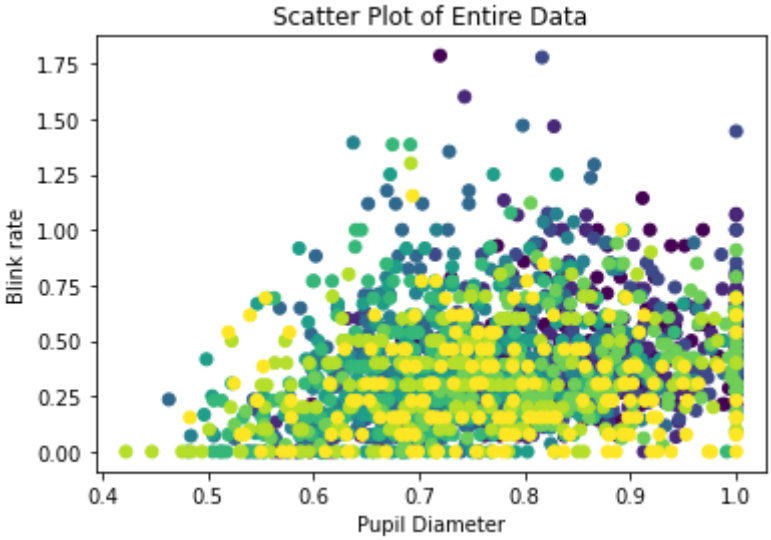
```
Requirement already satisfied: scikit-learn in f:\anaconda\lib\site-packages (0.24.
1)
Requirement already satisfied: scipy>=0.19.1 in f:\anaconda\lib\site-packages (from
scikit-learn) (1.6.2)
Requirement already satisfied: numpy>=1.13.3 in f:\anaconda\lib\site-packages (from
scikit-learn) (1.20.1)
Requirement already satisfied: joblib>=0.11 in f:\anaconda\lib\site-packages (from s
cikit-learn) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in f:\anaconda\lib\site-packages
(from scikit-learn) (2.1.0)
```

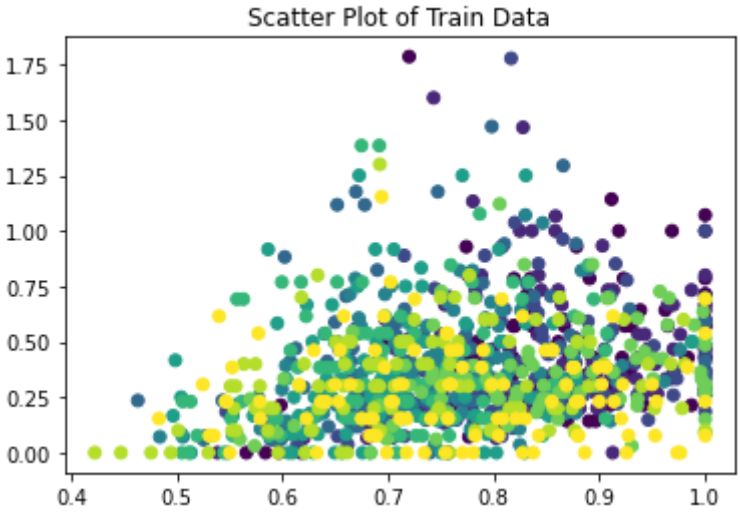
In [330…
```
import numpy
import matplotlib.pyplot as plt
import csv
import pandas as pd

###Path
pupildata_dir="E:\Sem 7\AM5011\project\ModelData\Pupil_data.csv"
blinkdata_dir="E:\Sem 7\AM5011\project\ModelData\BlinkData.csv"

###Load Data into DataFrame
pupil_data=pd.read_csv(pupildata_dir, header=None)
pupil_df=pd.DataFrame(pupil_data)
pupil_df.head()

blink_data=pd.read_csv(blinkdata_dir, header=None)
blink_df=pd.DataFrame(blink_data)
blink_df.head()

##Change the w value to the window you want the scatter plot for
w=10

plt.scatter(pupil_df.iloc[:,w-1],blink_df.iloc[:,w-1])
plt.title("Scatter Plot of each time window")
plt.xlabel("Pupil Diameter")
plt.ylabel("Blink rate")
plt.show()
```

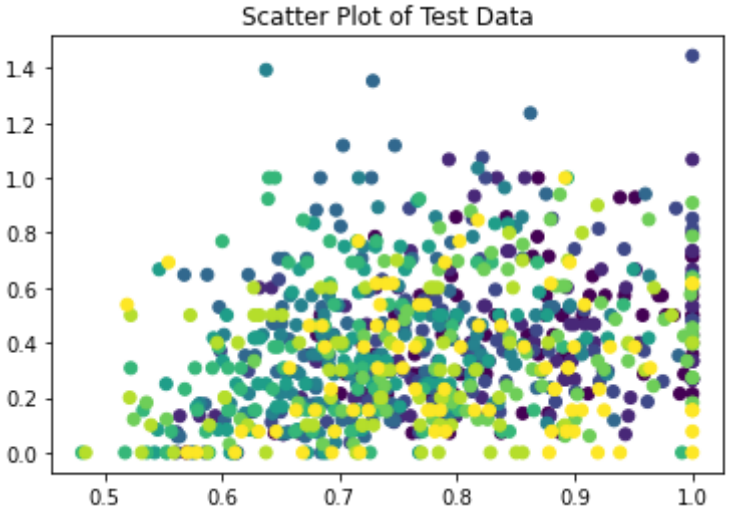


In [131…
```
###Reshaping data
##The dimensions of the table of the two parameters should be the same
```

```
#data_dir="E:\Sem 7\AM5011\project\ModelData\eye_data.csv"
data_mat=[]
train_mat=[]
test_mat=[]

for i in range(len(blink_df.columns)):
    for j in range(len(blink_df)):
        data_mat.append([i+1,pupil_df.iloc[j,i],blink_df.iloc[j,i]])
        if j<105:
            train_mat.append([i+1,pupil_df.iloc[j,i],blink_df.iloc[j,i]])
        else:
            test_mat.append([i+1,pupil_df.iloc[j,i],blink_df.iloc[j,i]])

train_df=pd.DataFrame(train_mat)
test_df=pd.DataFrame(test_mat)
data_df=pd.DataFrame(data_mat)

###Scatter Plot for the entire data
# plt.scatter(data_df.iloc[:,1],data_df.iloc[:,2],c=data_df.iloc[:,0])
# plt.title("Scatter Plot of Entire Data")
# plt.xlabel("Pupil Diameter")
# plt.ylabel("Blink rate")
# plt.show()

# plt.scatter(train_df.iloc[:,1],train_df.iloc[:,2],c=train_df.iloc[:,0])
# plt.title("Scatter Plot of Train Data")
# plt.xlabel("Pupil Diameter")
# plt.ylabel("Blink rate")
# plt.show()

plt.scatter(test_df.iloc[:,1],test_df.iloc[:,2],c=test_df.iloc[:,0])
plt.title("Scatter Plot of Test Data")
plt.xlabel("Pupil Diameter")
plt.ylabel("Blink rate")
plt.show()
```







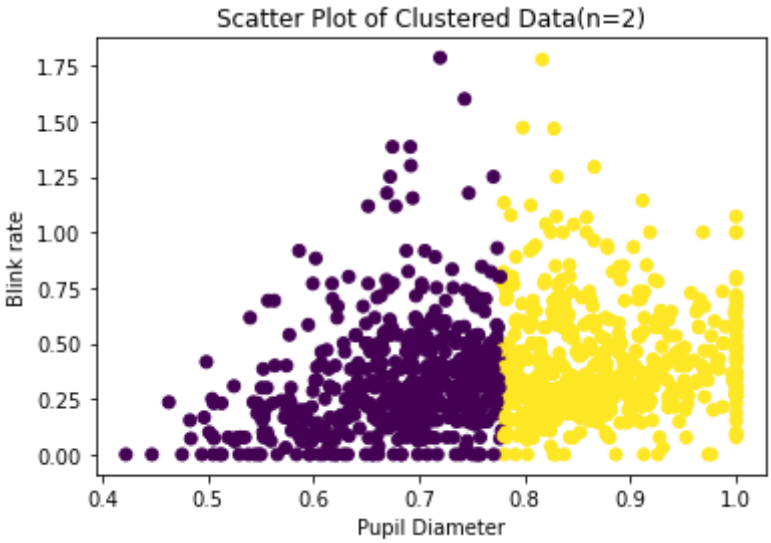
In [176…

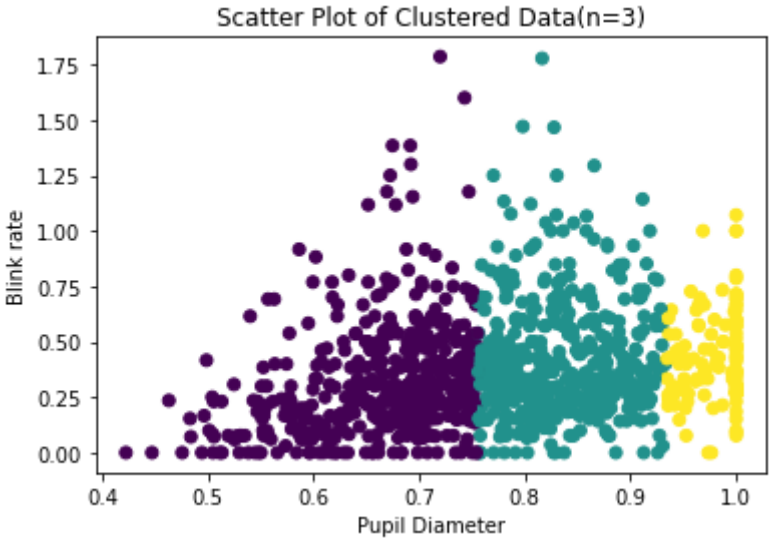
```
###K-Mean Clustering
##Implementation of elbow method

from sklearn.cluster import KMeans
from sklearn.cluster import Birch
# inertia=[]

# for i in range(1,10):
#     kmeans=KMeans(n_clusters=i)
#     kmeans.fit(train_df.iloc[:,1:2])
#     inertia.append(kmeans.inertia_)

# plt.plot(range(1,10), inertia, marker='o')
# plt.title('Elbow method')
# plt.xlabel('Number of clusters')
# plt.ylabel('Inertia')
# plt.show()
```

In [376…

```
###Training Data using KMeans

model_2= KMeans(n_clusters=2)
#model_2= Birch(threshold=0.01, n_clusters=2)
model_2.fit(train_df.iloc[:,1:2])

plt.scatter(train_df.iloc[:,1],train_df.iloc[:,2],c=model_2.labels_)
plt.title("Scatter Plot of Clustered Data(n=2)")
plt.xlabel("Pupil Diameter")
plt.ylabel("Blink rate")
```

```
plt.show()

#model_2= KMeans(n_clusters=3)
model_3= Birch(threshold=0.05, n_clusters=3)
model_3.fit(train_df.iloc[:,1:2])

plt.scatter(train_df.iloc[:,1],train_df.iloc[:,2],c=model_3.labels_)
plt.title("Scatter Plot of Clustered Data(n=3)")
plt.xlabel("Pupil Diameter")
plt.ylabel("Blink rate")
plt.show()
```





In [373…

```
train_class_2=[]
train_class_3=[]

for i in range(len(train_df)):
    train_class_2.append([train_df.iloc[i,0], model_2.labels_[i], train_df.iloc[i,1]
    train_class_3.append([train_df.iloc[i,0], model_3.labels_[i], train_df.iloc[i,1]

model_classes_2 = []
model_classes_3 = []

n_train=int(len(train_df)/10)   #Number of training samples

##Obtaining the number of the classes in each window
for i in range(10):
    one=0
    zero=0
```

```
    for j in range(n_train):
        if  model_2.labels_[i*n_train+j]==0:
            zero+=1
        else:
            one+=1
    model_classes_2.append([i+1, zero, one])

header_2=["Window", "Class 0", "Class 1"]
model_2_df=pd.DataFrame(model_classes_2, columns=header_2)

for i in range(10):
    one=0
    zero=0
    two=0
    for j in range(n_train):
        if  model_3.labels_[i*n_train+j]==0:
            zero+=1
        elif model_3.labels_[i*n_train+j]==1:
            one+=1
        else:
            two+=1
    model_classes_3.append([i+1, zero, one, two])

header_3=["Window", "Class 0", "Class 1","Class 2"]
model_3_df=pd.DataFrame(model_classes_3, columns = header_3)
```

In [374…

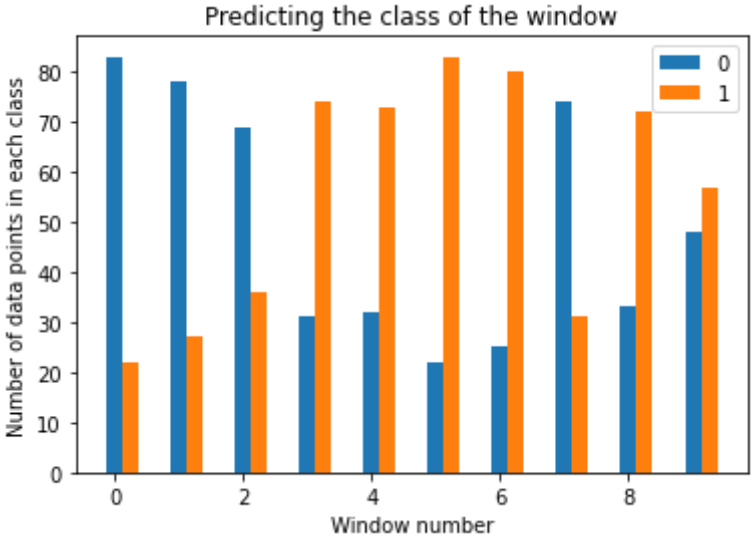
```
print(model_2_df)
print(model_3_df)
```

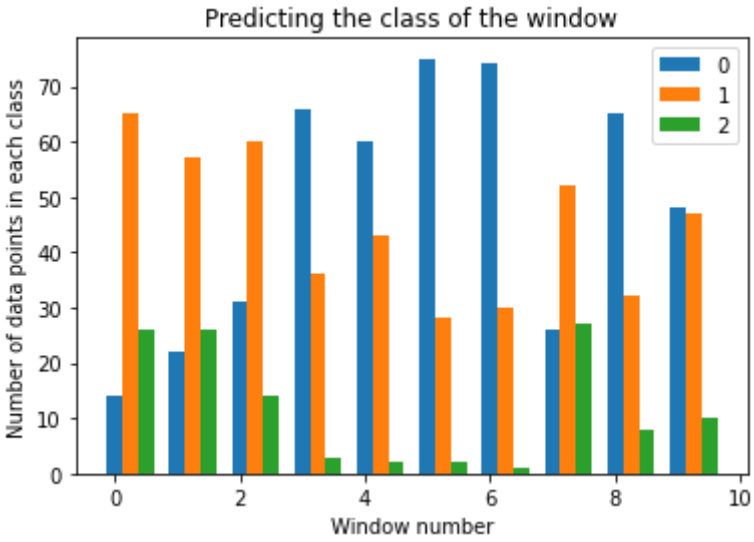
```
   Window  Class 0  Class 1
0       1       22       83
1       2       27       78
2       3       36       69
3       4       74       31
4       5       73       32
5       6       83       22
6       7       80       25
7       8       31       74
8       9       72       33
9      10       57       48
   Window  Class 0  Class 1  Class 2
0       1       14       65       26
1       2       22       57       26
2       3       31       60       14
3       4       66       36        3
4       5       60       43        2
5       6       75       28        2
6       7       74       30        1
7       8       26       52       27
8       9       65       32        8
9      10       48       47       10
```

In [368…

```
###Plotting the Train data classes according to the windows

barWidth = 0.25
br1 = numpy.arange(len(model_2_df))
br2 = [x + barWidth for x in br1]

plt.bar(br1,model_2_df.iloc[:,1], width=barWidth, label='0')
plt.bar(br2,model_2_df.iloc[:,2], width=barWidth, label='1')
plt.title("Predicting the class of the window")
plt.xlabel("Window number")
plt.ylabel("Number of data points in each class")
plt.legend()
```

```
plt.show()

barWidth = 0.25
br1 = numpy.arange(len(model_2_df))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]


plt.bar(br1,model_3_df.iloc[:,1], width=barWidth, label='0')
plt.bar(br2,model_3_df.iloc[:,2], width=barWidth, label='1')
plt.bar(br3,model_3_df.iloc[:,3], width=barWidth, label='2')
plt.title("Predicting the class of the window")
plt.xlabel("Window number")
plt.ylabel("Number of data points in each class")
plt.legend()
plt.show()
```





In [359…

```
###Classification of windows

#print(model_2_df)
train_clas=[[x[:][1]/n_train*100, x[:][2]/n_train*100, 0 , x[:][0]] for x in model_c

for i in range(10):
    if train_clas[i][0]>clas[i][1]:
        train_clas[i][2]= 0
    else:
        train_clas[i][2]=1
```

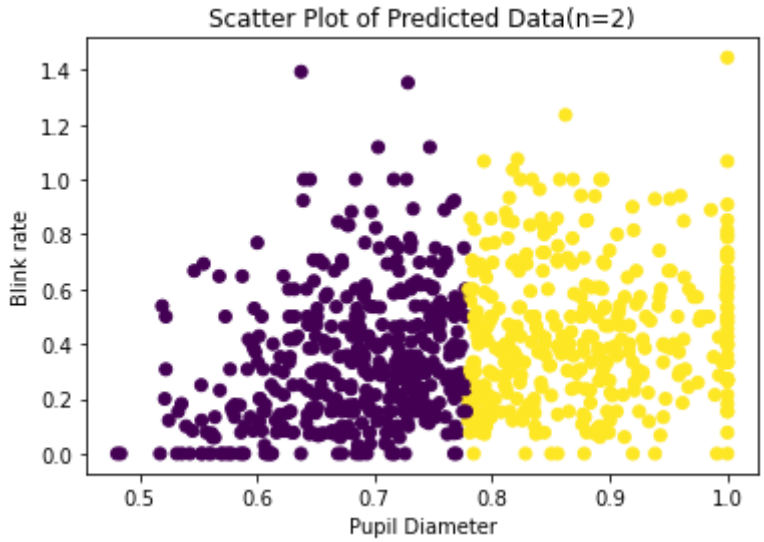
```
train_win=[]
for x in train_clas:
    print(x[3], x[2])
    train_win.append(x[2])
```

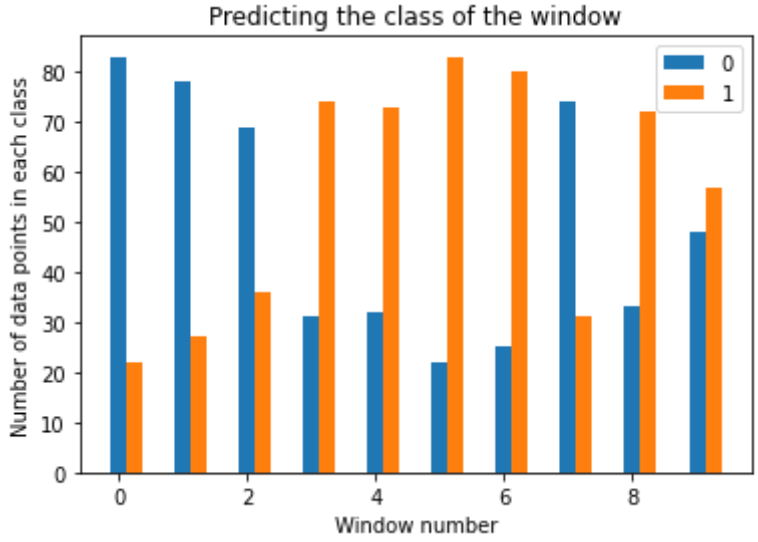
```
1 0
2 0
3 0
4 1
5 1
6 1
7 1
8 0
9 1
10 1
```

In [370…

```
###Testing Data
##The whole process above is repeated for the test data

test_model=model_2.predict(test_df.iloc[:,1:2])
plt.scatter(test_df.iloc[:,1],test_df.iloc[:,2],c=test_model)
plt.title("Scatter Plot of Predicted Data(n=2)")
plt.xlabel("Pupil Diameter")
plt.ylabel("Blink rate")
plt.show()

n_test = int(len(test_df)/10)
test_classes=[]

for i in range(10):
    one=0
    zero=0
    for j in range(n_test):
        if  test_model[i*n_test+j]==0:
            zero+=1
        else:
            one+=1
    test_classes.append([i+1, zero, one])

test_model_df=pd.DataFrame(model_classes_2, columns=header_2)

barWidth = 0.25
br1 = numpy.arange(len(test_model_df))
br2 = [x + barWidth for x in br1]

plt.bar(br1,test_model_df.iloc[:,1], width=barWidth, label='0')
plt.bar(br2,test_model_df.iloc[:,2], width=barWidth, label='1')
plt.title("Predicting the class of the window")
plt.xlabel("Window number")
plt.ylabel("Number of data points in each class")
plt.legend()
plt.show()

test_clas=[[x[1]/n_test*100, x[2]/n_test*100, 0 , x[0]] for x in test_classes]

for i in range(10):
    if test_clas[i][0]>clas[i][1]:
        test_clas[i][2]= 0
    else:
        test_clas[i][2]=1

test_win=[]
print("Test data window and class")
for x in test_clas:
```

```
    print(x[3], x[2])
    test_win.append(x[2])
```






```
Test data window and class
1 0
2 0
3 0
4 0
5 1
6 1
7 0
8 1
9 1
10 1
```

In [365…

```
###Comparing the classes of windows from training and testing
acc=[]

for i in range(10):
    acc.append(train_win==test_win)

val=acc.count(True)
val_perc = val/10*100
print(val_perc)
```

```
100.0
```