Name: pasala neelima

RegNo: 24MDT1064

Experiment : 05 & 06

Task : confusion matrix & regression and k-means & db-scan.

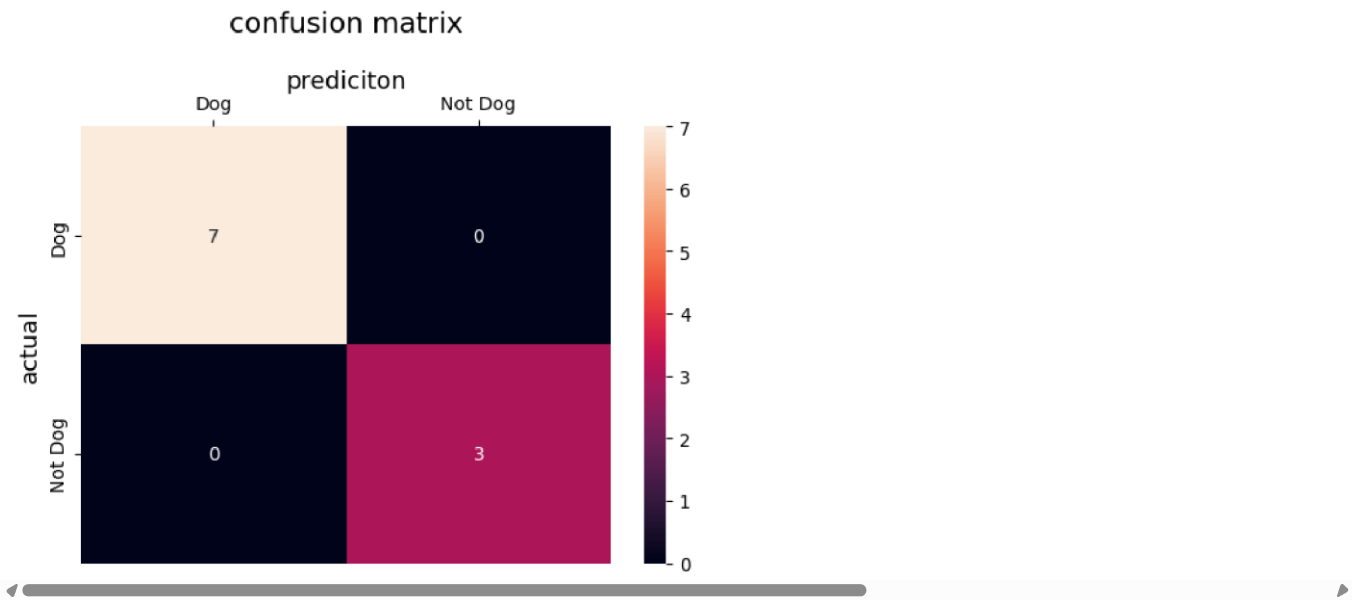
```
#binary classification import numpy as np from sklearn.metrics
import confusion_matrix,classification_report import seaborn as
sns import matplotlib.pyplot as plt


#class labels actual=np.array(['Dog','Dog','Dog','Not Dog','Dog','Not
Dog','Dog','Dog','Dog','Not Dog']) predicted=np.array(['Dog','Dog','Dog','Not Dog','Dog','Not
Dog','Dog','Dog','Dog','Not Dog'])


#confusion matrix
cm=confusion_matrix(actual,predicted)
print(cm)
```

```
[[7 0]
 [0 3]]
```

```
#heatmap of confusion matrix sns.heatmap(cm,annot=True, xticklabels=['Dog','Not Dog'] ,
yticklabels=['Dog','Not Dog']) plt.gca().xaxis.set_label_position('top')
plt.xlabel('prediciton',fontsize=13) plt.gca().xaxis.tick_top()
plt.ylabel('actual',fontsize=13) plt.title('confusion matrix',fontsize=15,pad=20)
plt.gca().figure.subplots_adjust(bottom=0.2) plt.show()
```



Start coding or generate with AI.

```
#classficiation report

print(classification_report(actual,predicted))
```

```
precision    recall  f1-score   support

           Dog       1.00      1.00      1.00         7
     Not Dog       1.00      1.00      1.00         3

    accuracy                           1.00        10
   macro avg       1.00      1.00      1.00        10
weighted avg       1.00      1.00      1.00        10
```

```
#multiclass claassfication
import numpy as np
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classification_report
import matplotlib.pyplot as plt


#actual lables y_true = ['cat']*10 + ['Dog']*12
+ ['Horse']*10 print(y_true)
```

```
['cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'cat', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog
```

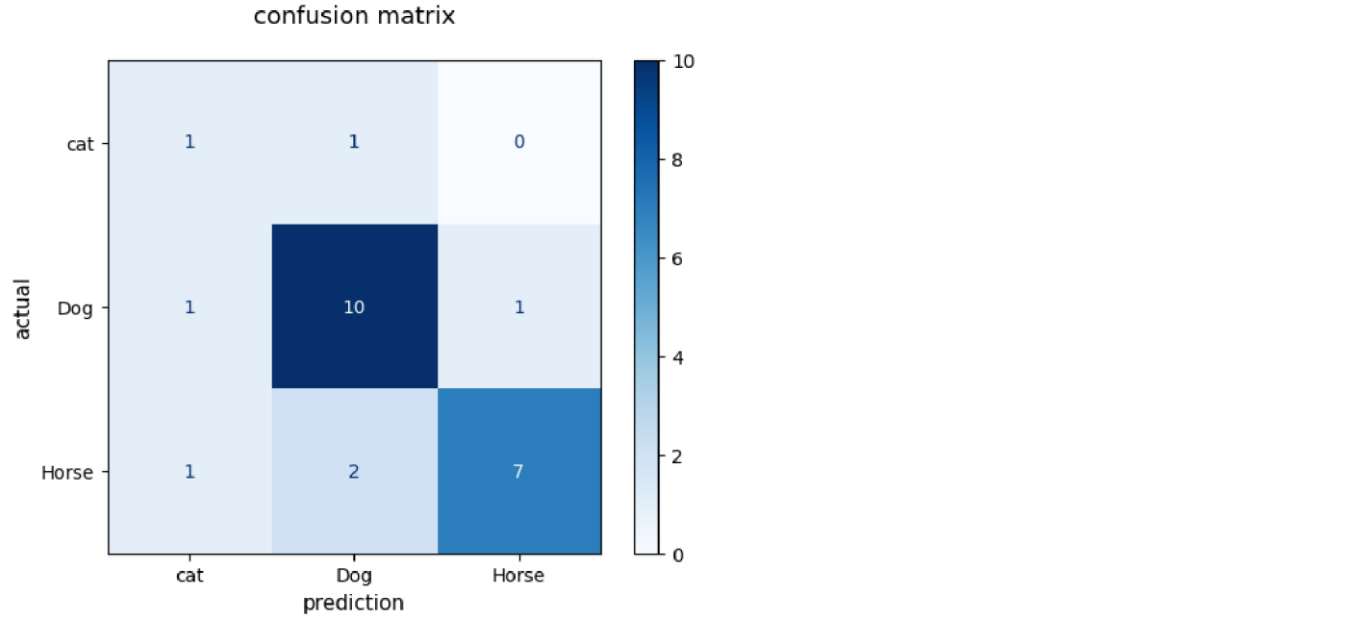
```
#predicaTED TOOLS y_pred = ['Cat']*8 + ['Dog'] +
['cat']*2+['Dog']*10+['Horse']*8+['Dog']*2 print(y_pred)
```

```
['Cat', 'Cat', 'Cat', 'Cat', 'Cat', 'Cat', 'Cat', 'Cat', 'Dog', 'cat', 'cat', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog', 'Dog
```

```
#classes classes =
['cat','Dog','Horse']
```

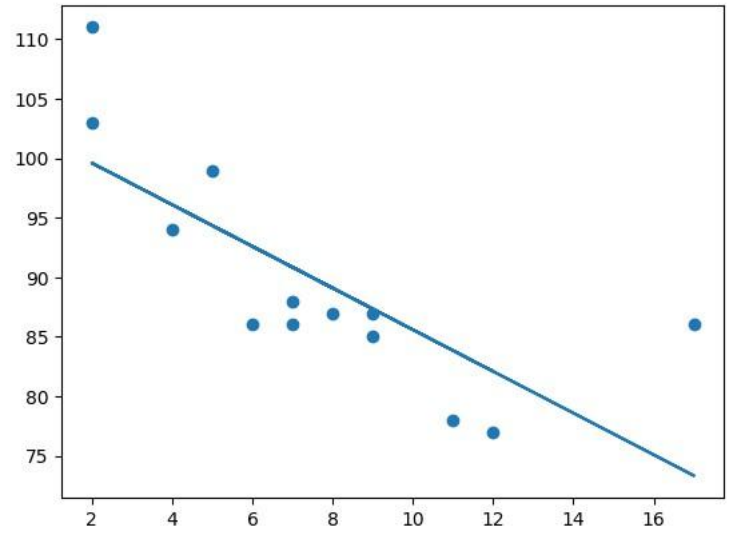
```
#generate the confusion matrix cm =
confusion_matrix(y_true,y_pred,labels=classes)
disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=classes)
disp.plot(cmap=plt.cm.Blues) plt.title('confusion
matrix',fontsize=13,pad=20) plt.xlabel('prediction',fontsize=11)
plt.ylabel('actual',fontsize=11)
```

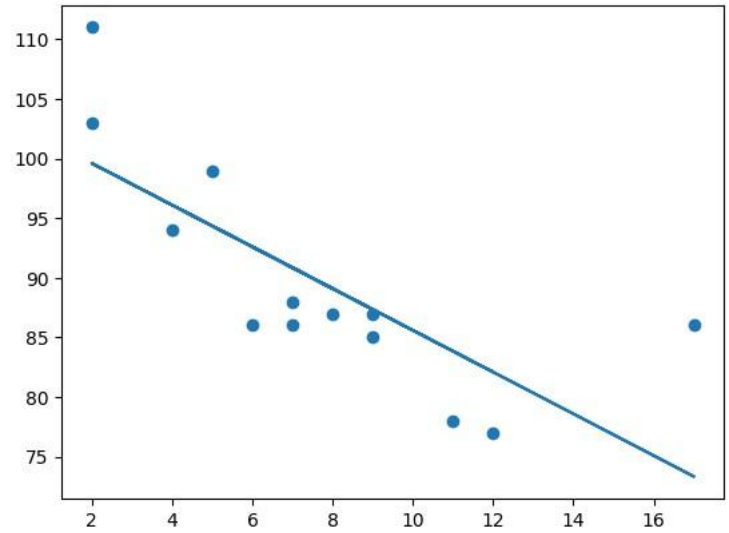
```
Text(0, 0.5, 'actual')
```



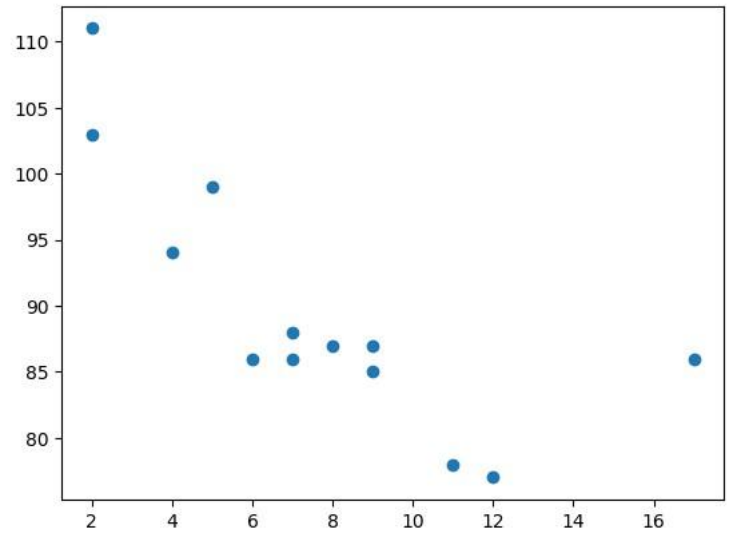

```
#linear regression import matplotlib.pyplot as plt import
scipy.stats as stats x = [5,7,8,7,2,17,2,9,4,11,12,9,6] y =
[99,86,87,88,111,86,103,87,94,78,77,85,86] slope ,
intercept , r , p , std_err = stats.linregress(x,y) def
myfunc(x):
    return slope * x + intercept mymodel
= list(map(myfunc,x)) print(mymodel)
print('correlation coefficient:',r) yhat
= myfunc(10) print('predicted value at
x=10 is',yhat) plt.scatter(x,y)
plt.plot(x, mymodel) plt.show()
plt.scatter(x,y) plt.plot(x, mymodel)
plt.show() plt.scatter(x,y)
```

```
[94.3495217071376, 90.84694628403238, 89.09565857247976, 90.84694628403238, 99.60338484179543, 73.33406916850626, 99.60338484179543,
correlation coefficient: -0.758591524376155
predicted value at x=10 is 85.59308314937454
```





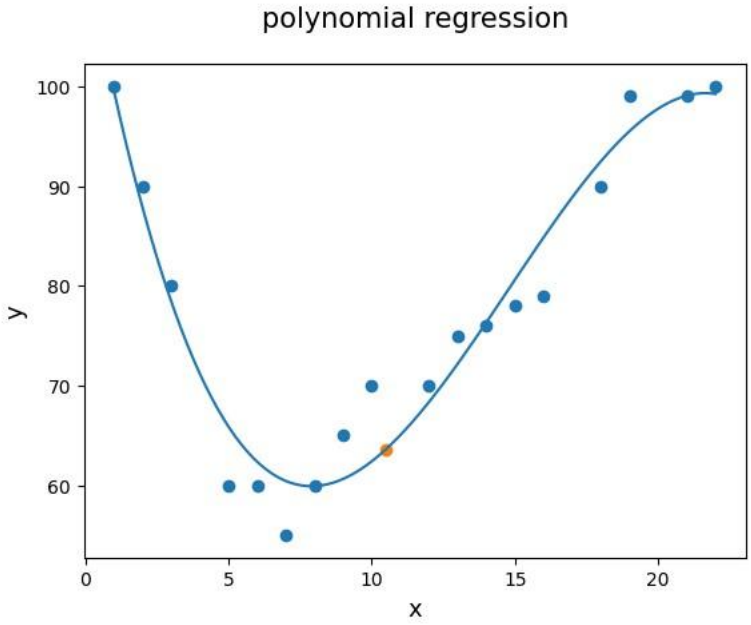

```
<matplotlib.collections.PathCollection at 0x7e72c1b22c10>
```




```
#polynomial regression import numpy import matplotlib.pyplot
as plt x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22] y
= [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]
mymodel = numpy.poly1d(numpy.polyfit(x,y,3)) yhat =
mymodel(10.5) print('predicted value at x=10.5 is',yhat)
myline = numpy.linspace(1,22,100) plt.scatter(x,y)
plt.scatter(10.5,yhat)
plt.plot(myline,mymodel(myline)) plt.xlabel('x'
,fontsize=13) plt.ylabel('y',fontsize=13)
```

```
plt.title('polynomial regression',fontsize=15,pad=20)
plt.show()
```

```
predicted value at x=10.5 is 63.61798136290647
```



Start coding or generate with AI.

Name: pasala neelima

RegNo: 24MDT1064

LO-U5+U6 LAB-8

```
import pandas as pd import numpy as np import
matplotlib.pyplot as plt from sklearn.cluster
import KMeans from sklearn.preprocessing import
StandardScaler

# Load the dataset df =
pd.read_csv('/content/driver-data.csv')
```

```
X = df[['mean_dist_day', 'mean_over_speed_perc']].values

# Standardize the data scaler =
StandardScaler() X_scaled =
scaler.fit_transform(X)
```

```
import pandas as pd import numpy as np import
matplotlib.pyplot as plt from sklearn.cluster
import DBSCAN from sklearn.preprocessing import
StandardScaler

# Load the dataset
df = pd.read_csv('/content/driver-data.csv')
X = df[['mean_dist_day', 'mean_over_speed_perc']].values

# Standardize the data
scaler = StandardScaler()
```

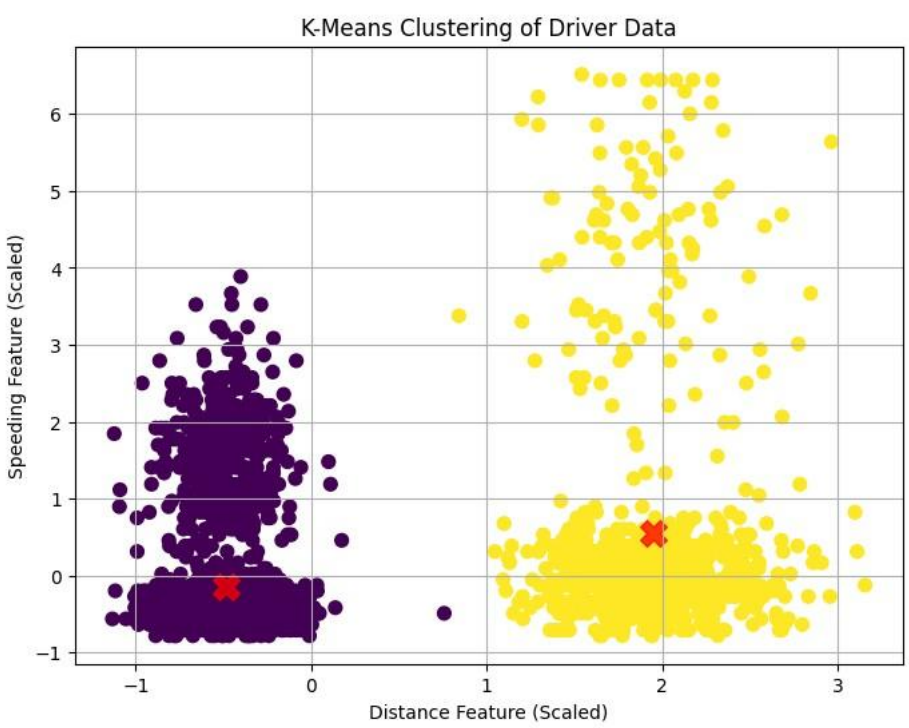
```
X_scaled = scaler.fit_transform(X)
```



```
# Apply K-Means kmeans = KMeans(n_clusters=2, random_state=42) #
Assuming 2 clusters y_kmeans = kmeans.fit_predict(X_scaled)

# Plot the results plt.figure(figsize=(8, 6)) plt.scatter(X_scaled[:, 0],
X_scaled[:, 1], c=y_kmeans, s=50, cmap='viridis') centers =
kmeans.cluster_centers_ plt.scatter(centers[:, 0], centers[:, 1], c='red',
s=200, alpha=0.75, marker='X') plt.title('K-Means Clustering of Driver Data')
plt.xlabel('Distance Feature (Scaled)') plt.ylabel('Speeding Feature (Scaled)')
plt.grid()
```

```
plt.show()
```



```
# Apply DBSCAN dbscan = DBSCAN(eps=0.5, min_samples=5) # Adjust eps and
min_samples as needed y_dbscan = dbscan.fit_predict(X_scaled)
```

```
# Plot the results plt.figure(figsize=(8, 6)) plt.scatter(X_scaled[:, 0],
X_scaled[:, 1], c=y_dbscan, s=50, cmap='viridis') plt.title('DBSCAN
Clustering of Driver Data') plt.xlabel('Distance Feature (Scaled)')
plt.ylabel('Speeding Feature (Scaled)') plt.grid()
plt.show()
```