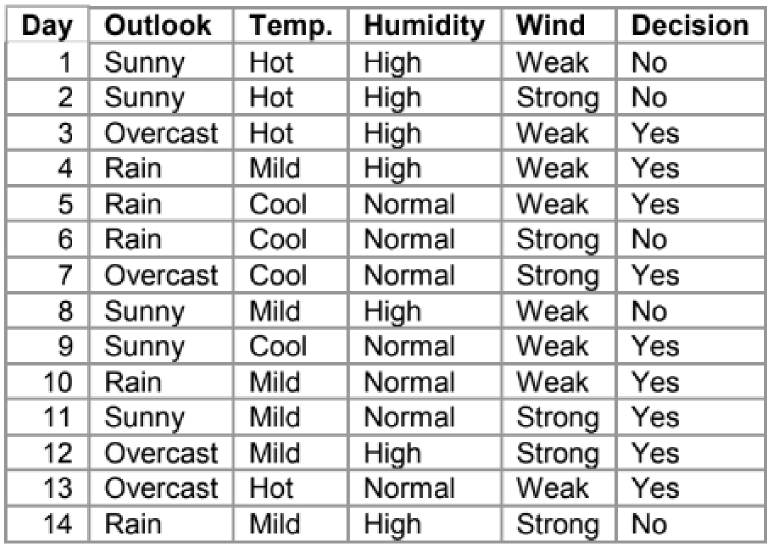
Machine learning LAB-09

Perumalla Dharan
AP21110010201

1)

Implement Naïve Bayes classifier for following datasets and evaluate the classification performance. Draw the confusion matrix, compute accuracy, error and other measures as applicable.

a. The enjoy sports dataset as. given below

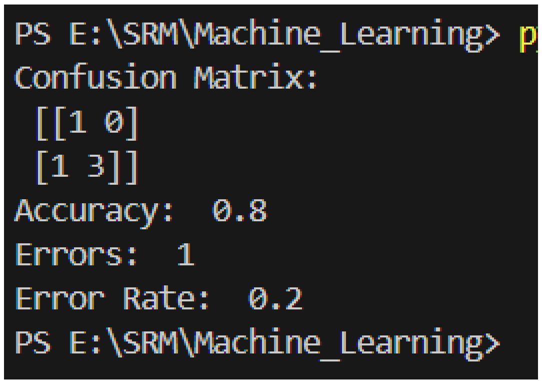


Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,
accuracy_score

# Define the dataset
data = {
    'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rain',
'Rain', 'Rain', 'Overcast', 'Sunny', 'Sunny', 'Rain',
'Sunny', 'Overcast', 'Overcast', 'Rain'],
    'Temp': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool',
'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot',
'Mild'],
    'Humidity': ['High', 'High', 'High', 'High', 'Normal',
'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal',
'High', 'Normal', 'High'],
    'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak',
'Strong', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong',
'Strong', 'Weak', 'Strong'],
    'Decision': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No',
'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
}

# Convert categorical variables into numerical ones
le = LabelEncoder()
data_encoded = {col: le.fit_transform(data[col]) for col in
data}

# Split the dataset into training and testing sets
X = list(zip(data_encoded['Outlook'], data_encoded['Temp'],
data_encoded['Humidity'], data_encoded['Wind']))
y = data_encoded['Decision']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)

# Implement the Naive Bayes classifier
model = GaussianNB()

# Train the classifier with the training set
model.fit(X_train, y_train)

# Predict the test set results
y_pred = model.predict(X_test)
```

```
# Evaluate the model
print("Confusion Matrix:\n", confusion_matrix(y_test,
y_pred))
print("Accuracy: ", accuracy_score(y_test, y_pred))

# Print the errors
errors = 0
for i in range(len(y_test)):
    if y_test[i] != y_pred[i]:
        errors += 1
print("Errors: ", errors)
print("Error Rate: ", errors / len(y_test))
```

<u>Output-</u>



```
PS E:\SRM\Machine_Learning> p
Confusion Matrix:
 [[1 0]
 [1 3]]
Accuracy:  0.8
Errors:  1
Error Rate:  0.2
PS E:\SRM\Machine_Learning>
```

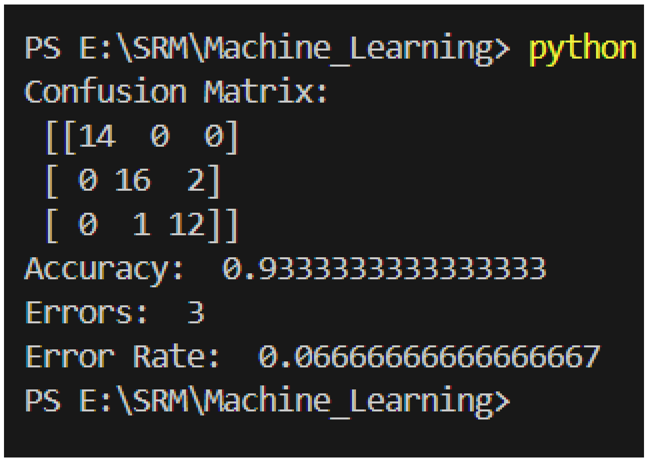
---

b. The Iris dataset

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,
accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)

# Implement the Naive Bayes classifier
model = GaussianNB()

# Train the classifier with the training set
model.fit(X_train, y_train)

# Predict the test set results
y_pred = model.predict(X_test)

# Evaluate the model
print("Confusion Matrix:\n", confusion_matrix(y_test,
y_pred))
print("Accuracy: ", accuracy_score(y_test, y_pred))

# Print the errors
errors = 0
for i in range(len(y_test)):
    if y_test[i] != y_pred[i]:
        errors += 1
print("Errors: ", errors)
print("Error Rate: ", errors / len(y_test))
```

Output-



```
PS E:\SRM\Machine_Learning> python
Confusion Matrix:
 [[14  0  0]
 [ 0 16  2]
 [ 0  1 12]]
Accuracy:  0.9333333333333333
Errors:  3
Error Rate:  0.06666666666666667
PS E:\SRM\Machine_Learning>
```