Machine Learning Lab Sheet 03

In this lab, you will explore two fundamental tasks in machine learning: regression and classification. You will use Python in Google Colab to implement and compare simple linear regression, decision tree regression, naive Bayes classification, and decision tree classification.

Part 1: Regression Task

Objective: Implement regression models and evaluate their performance.

Task 1.1: Simple Linear Regression

- Import necessary libraries.
- Load a dataset suitable for simple linear regression (e.g., a dataset with two numerical variables).
- Split the data into training and testing sets.
- Implement and train a simple linear regression model.
- Evaluate the model's performance using appropriate regression metrics (e.g., Mean Squared Error, R-squared).
- Visualize the regression line.

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the California Housing dataset
california_housing = fetch_california_housing()
data = pd.DataFrame(data=california_housing.data,
columns=california_housing.feature_names)
target = pd.DataFrame(data=california_housing.target, columns=['MEDV'])

# Select one feature (e.g., 'Rooms' - average number of rooms) for simple
linear regression
X = data[['AveRooms']]
y = target['MEDV']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Implement and train a simple linear regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regressor.predict(X_test)

# Evaluate the model's performance using Mean Squared Error and R-squared
mse = mean_squared_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")

# Visualize the regression line
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicted')
plt.xlabel('Average Number of Rooms (AveRooms)')
plt.ylabel('Median Value (MEDV)')
plt.title('Simple Linear Regression')
plt.legend()
plt.show()
```

Task 1.2: Decision Tree Regression

Import necessary libraries (`from sklearn.tree import DecisionTreeRegressor`).

- Load the same dataset used in Task 1.1.
- Implement and train a decision tree regression model.
- Evaluate the model's performance using regression metrics.
- Visualize the regression tree.

Part 2: Classification Task

Objective: Implement classification models and assess their accuracy.

Task 2.1: Naive Bayes Classification

- Import necessary libraries.
- Load a dataset suitable for classification (e.g., the Iris dataset).
- Split the data into training and testing sets.
- Implement and train a Naive Bayes classification model.
- Evaluate the model's accuracy using appropriate classification metrics (e.g., accuracy, confusion matrix).
- Visualize the classification results.

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns

# Load the Iris dataset
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

```
target = pd.DataFrame(data=iris.target, columns=['Species'])

# Split the data into training and testing sets
X = data
y = target['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Implement and train a Naive Bayes classification model (Gaussian Naive
Bayes)
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Generate and visualize the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Task 2.2: Decision Tree Classification

1. Import necessary libraries.
2. Load the same dataset used in Task 2.1.
3. Implement and train a decision tree classification model.
4. Evaluate the model's accuracy using classification metrics.
5. Visualize the classification tree.

Homework Tasks:

Choose a real-world dataset and perform a regression task using both simple linear regression and decision tree regression. Compare the model performance. Choose a different real-world dataset and perform a classification task using both Naive Bayes and decision tree classification. Compare the model accuracy.
Write a brief report summarizing your findings, including the dataset used, data preprocessing steps, model implementation, evaluation metrics, and a discussion of the results.