Machine Learning Lab Sheet 04

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 SVM for regression, SVM for classification, and logistic regression.

Part 1: Regression Task

Objective: Implement regression models and evaluate their performance.

Task 1.1: Support vector Machine for 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.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Create a sample dataset with two numerical features (X) and a target
variable (y)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel()

# Add noise to the target variable
y[::5] += 3 * (0.5 - np.random.rand(16))

# 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 Support Vector Machine for Regression (SVR) model
svr = SVR(kernel='linear', C=1)
svr.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svr.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('Feature (X)')
plt.ylabel('Target (y)')
plt.title('Support Vector Machine Regression')
plt.legend()
plt.show()
```

use the previous code to solve the regression task for diabetes dataset

```
from sklearn import datasets
# Load the diabetes dataset
diabetes = datasets.load_diabetes()
# Access data and target variables
X = diabetes.data  # Features
y = diabetes.target  # Target variable
# Now, you can start your regression task using X and y
```

Part 2: Classification Task

Objective: Implement classification models and assess their accuracy.

Task 2.1: SVM for 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
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

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

# 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 an SVM classifier
svm_classifier = SVC(kernel='linear', C=1)
svm_classifier.fit(X_train, y_train)
```

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

# Evaluate the model's accuracy and provide a classification report
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred,
target_names=iris.target_names)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(report)
```

Task 2.2: Logistic Regression for 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.

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

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

# 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 Logistic Regression classifier
logistic_classifier = LogisticRegression()
logistic_classifier.fit(X_train, y_train)

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

# Evaluate the model's accuracy and provide a classification report
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred,
target_names=iris.target_names)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(report)
```

Homework Tasks:

Choose a real-world dataset and perform a classification task using both SVM and logistic regression. Compare the model performance.

Write a brief report summarizing your findings, including the dataset used, data preprocessing steps, model implementation, evaluation metrics, and a discussion of the results.