HYDERABAD INSTITUTE OF ARTS, SCIENCE, AND TECHNOLOGY

Artificial Intelligence Lab -4
Instructor: Ayesha Eman
Date: 17/10/2025

Lab Title:

Regression Algorithms in Machine Learning

Objective:

The objective of this lab is to understand and implement regression algorithms for predicting continuous outcomes. Students will apply Linear Regression and Random Forest Regression on a real-world student dataset to analyze performance, evaluate predictive capability, and compare model effectiveness using standard evaluation metrics.

1. Regression in Machine Learning

Regression is a supervised learning technique used to predict continuous numerical values based on input features. It is widely used in applications such as predicting grades, prices, or sales trends. Unlike classification, which deals with categories, regression focuses on estimating relationships between variables and predicting numeric outcomes.

2. Regression Algorithms Overview

2.1 Linear Regression

Linear Regression is the simplest regression technique that models a straight-line relationship between input features and a target variable. It serves as a baseline model, providing a foundation to understand how input factors affect the output variable. However, it performs best when the relationship between data variables is linear.

2.2 Random Forest Regression

Random Forest Regression is an ensemble method that builds multiple decision trees and averages their results for more accurate predictions. It is highly effective in handling non-

linear relationships, noisy data, and feature interactions, often outperforming simpler models in real-world scenarios.

3. Implementation Using Python

```
# Import required libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Load Dataset
data = pd.read_excel('students_dataset.xlsx')

# Drop irrelevant columns
data = data.drop(['Name'], axis=1)

# Convert numeric columns
numeric_cols = ['Marks', 'Hours_Studied', 'Attendance']
for col in numeric_cols:
    data[col] = pd.to_numeric(data[col], errors='coerce')

# Fill missing values
data = data.fillna(data.median(numeric_only=True))

# Encode categorical columns
le = LabelEncoder()
for col in data.select_dtypes(include='object').columns:
    data[col] = le.fit_transform(data[col].astype(str))

# Step 2: Define Features and Target
X = data.drop('Marks', axis=1)
y = data['Marks']

# Step 3: Split Dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Feature Scaling (important for Linear Regression)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 5: Train Models
# Linear Regression
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
lr_pred = lr.predict(X_test_scaled)

# Random Forest Regression
rf = RandomForestRegressor(n_estimators=150, random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)

# Step 6: Evaluation Function
def evaluate_model(y_true, y_pred, model_name):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    print(f"=== {model_name} Evaluation ===")
    print(f"Mean Squared Error: {mse:.2f}")
    print(f"Root Mean Squared Error: {rmse:.2f}")
    print(f"Mean Absolute Error: {mae:.2f}")
    print(f"R² Score: {r2:.4f}\n")
    return mse, rmse, mae, r2

lr_scores = evaluate_model(y_test, lr_pred, "Linear Regression")
rf_scores = evaluate_model(y_test, rf_pred, "Random Forest Regression")

# Step 7: Visualization (Line Plots Only)
plt.figure(figsize=(12,5))

# Linear Regression Line Plot
plt.subplot(1,2,1)
plt.plot(y_test.values, label='Actual Marks', color='blue')
plt.plot(lr_pred, label='Predicted Marks', color='red')
plt.title("Linear Regression: Actual vs Predicted Marks")
plt.xlabel("Samples")
plt.ylabel("Marks")
plt.legend()
```

```
# Random Forest Regression Line Plot
plt.subplot(1,2,2)
plt.plot(y_test.values, label='Actual Marks', color='blue')
plt.plot(rf_pred, label='Predicted Marks', color='green')
plt.title("Random Forest Regression: Actual vs Predicted Marks")
plt.xlabel("Samples")
plt.ylabel("Marks")
plt.legend()

plt.tight_layout()
plt.show()

# Step 8: Model Comparison
comparison = pd.DataFrame({
   'Model': ['Linear Regression', 'Random Forest Regression'],
   'R² Score': [lr_scores[3], rf_scores[3]],
   'RMSE': [lr_scores[1], rf_scores[1]],
   'MAE': [lr_scores[2], rf_scores[2]]
})
print("\nModel Comparison:\n", comparison)
```

4. Evaluation Metrics

Mean Squared Error (MSE)	Measures average squared difference between actual and predicted values.
Root Mean Squared Error (RMSE)	Indicates how close predictions are to the real values, in the same unit as the target.
Mean Absolute Error (MAE)	Shows average absolute deviation between predictions and true values.
R^2 Score	Represents how well the model explains variation in the data (closer to 1 indicates a better fit).

5. Visualization and Insights

The scatter plots display how accurately each model predicts the actual marks. Linear Regression works effectively when relationships between variables are direct and proportional. Random Forest Regression captures complex interactions and often produces higher prediction accuracy. The comparison table provides a clear summary of model performance using objective metrics.

6. Discussion Questions

1. Compare the performance of Linear Regression and Random Forest Regression in terms of accuracy and interpretability.

2. Why is feature scaling important for Linear Regression but less critical for Random Forest Regression?

3. Discuss situations where a linear model might fail to capture real-world behavior.

4. How does Random Forest reduce overfitting compared to a single decision tree?

5. What insights can be drawn from the R^2 Score in evaluating regression models?

6. Suggest additional preprocessing techniques that could further improve regression performance.

7. Lab Tasks

Students are required to perform the following tasks to reinforce their understanding of both classification and regression models implemented in the Artificial Intelligence labs.

1. Task 1: Implement a Logistic Regression model using your own small dataset and compare its accuracy with Random Forest Classifier.
2. Task 2: Modify the classification code to handle multi-class classification (more than two output categories).
3. Task 3: Apply both Linear and Random Forest Regression models on a different dataset (e.g., salary prediction or housing prices) and analyze performance using R^2 and RMSE.