Department of Information Engineering Technology

The Superior University, Lahore

Artificial Intelligence Lab

Experiment No.3

Supervised Learning: Linear Regression

Prepared for

_____

By:

Name: _____

ID: _____

Section: _____

Semester: _____

Total Marks: _____

Obtained Marks: _____

Signature: _____

Date: _____

Experiment No. 3

Supervised Learning: Linear Regression

(Rubrics)

Name: _____ **Roll No. : _____**

A. PSYCHOMOTOR

Sr. No.	Criteria	Allocated Marks	Unacceptable 0%	Poor 25%	Fair 50%	Good 75%	Excellent 100%	Total Obtained
1	Follow Procedures	1	0	0.25	0.5	0.75	1	
2	Software and Simulations	2	0	0.5	1	1.5	2	
3	Accuracy in Output Results	3	0	0.75	1.5	2.25	3	
Sub Total		**6**	**Sub Total marks Obtained in Psychomotor(P)**					

B. AFFECTIVE

Sr. No.	Criteria	Allocated Marks	Unacceptable 0%	Poor 25%	Fair 50%	Good 75%	Excellent 100%	Total Obtained
1	Respond to Questions	1	0	0.25	0.5	0.75	1	
2	Lab Report	1	0	0.25	0.5	0.75	1	
3	Assigned Task	2	0	0.5	1	1.5	2	
Sub Total		**4**	**Sub Total marks Obtained in Affective (A)**					

Instructor Name:_____ **Total Marks (P+A): _____**

Instructor Signature:_____ **Obtained Marks (P+A): _____**

Introduction to Machine Learning

Machine learning is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to learn and make predictions or decisions without being explicitly programmed for specific tasks. In other words, machine learning systems can improve their performance on a task through experience or data.

Here are some key concepts and components of machine learning:

Data: Machine learning algorithms require data to learn from. This data can be labeled (with known outcomes) or unlabeled (without known outcomes). The quality and quantity of data often play a crucial role in the success of a machine learning model.

Features: Features are the characteristics or attributes extracted from the data that the model uses to make predictions. Feature engineering is the process of selecting, transforming, and preprocessing these attributes to make them suitable for machine learning.

Algorithms: Machine learning algorithms are the mathematical and statistical models that learn patterns and relationships in data. There are 3 types of machine learning algorithms, including:

- **Supervised learning:** Models are trained on labeled data to make predictions or classify data points into predefined categories.
- **Unsupervised learning:** Models discover patterns and structures in unlabeled data, often used for clustering and dimensionality reduction.
- **Reinforcement learning:** Agents learn to make decisions by interacting with an environment and receiving rewards or penalties.

Training: Training a machine learning model involves feeding it a dataset and allowing it to adjust its internal parameters to minimize a predefined objective function, such as error or loss. The model iteratively improves its performance during this process.

Testing and Evaluation: After training, machine learning models are tested on new, unseen data to evaluate their generalization performance. Common evaluation metrics include accuracy, precision, recall, F1-score, and mean squared error, among others.

Hyperparameter Tuning: Fine-tuning the hyperparameters of a machine learning model is an essential step to optimize its performance. Hyperparameters are settings that are not learned from data but are set before training and can significantly affect a model's behavior.

Deployment: Once a machine learning model is trained and evaluated, it can be deployed in real-world applications to make predictions or assist in decision-making. This might involve integrating the model into a software application or system.

Introduction to Supervised Learning

Supervised learning is a type of machine learning approach in artificial intelligence (AI) where the algorithm learns from labeled data to make predictions or decisions. It is one of the most common and fundamental paradigms in machine learning and is widely used in various applications, including image recognition, natural language processing, recommendation systems, and more.

Here's how supervised learning works:

Data Collection: You start by collecting a dataset that consists of input-output pairs. Each data point in the dataset includes both input features (the data you use to make predictions) and the corresponding target output (the desired prediction or label).

Data Labeling: In supervised learning, the target output is labeled. For example, in a classification problem, the labels could be categories like "cat," "dog," or "car." In a regression problem, the labels are typically numeric values.

Training: The labeled dataset is split into two subsets: a training set and a test set. The training set is used to train the machine learning model. During training, the model learns the underlying patterns and relationships between the input features and the target output by adjusting its internal parameters.

Model Selection: You choose a specific machine learning algorithm or model architecture that is suitable for your problem. Common algorithms include decision trees, support vector machines, neural networks, and many others.

Training Process: The algorithm iteratively processes the training data, making predictions and comparing them to the actual labels. It uses a loss or error function to measure the difference between its predictions and the true labels. The model's parameters are adjusted to minimize this error.

Validation: Periodically, you evaluate the model's performance on the validation set (which is separate from the test set) to ensure it is not overfitting (memorizing the training data) and is learning generalizable patterns.

Testing: After training, you use the test set, which the model has never seen before, to assess its performance on new, unseen data. This step helps you gauge how well the model will perform in real-world scenarios.

Deployment: If the model performs well on the test set and meets the desired accuracy and reliability requirements, it can be deployed in real applications to make predictions on new, unlabeled data.

Supervised learning is versatile and can be applied to various types of problems, including:

- **Classification:** Assigning input data to predefined categories or labels. For example, spam email detection, image classification, or sentiment analysis.
- **Regression:** Predicting continuous numeric values. For example, predicting house prices, stock prices, or temperature.
- **Object Detection:** Identifying and locating objects within an image or video.
- **Speech Recognition:** Converting spoken language into text.
- **Recommendation Systems:** Recommending products, movies, or content based on user preferences.
- **Natural Language Processing (NLP):** Tasks such as text classification, machine translation, and text generation.

Introduction to Linear Regression

Linear regression is a fundamental statistical and machine learning technique used for modeling and analyzing the relationship between a dependent variable and one or more independent variables. It is a type of supervised learning algorithm, often applied in predictive modeling and data analysis tasks.

Here are the key components and characteristics of linear regression:

Dependent Variable (Target): Linear regression aims to predict or explain a dependent variable (also known as the target or response variable). This is the variable you want to understand or make predictions about. In mathematical notation, the dependent variable is typically denoted as "Y."

Independent Variables (Predictors or Features): Linear regression considers one or more independent variables (also known as predictors or features) that are believed to influence or explain variations in the dependent variable. In mathematical notation, the independent variables are typically denoted as "X."

Linear Relationship: Linear regression assumes that there is a linear relationship between the independent variables and the dependent variable. In simple linear regression (with one independent variable), this relationship is represented by a straight line. In multiple linear regression (with multiple independent variables), it's represented by a hyperplane.

Linear Equation: The linear relationship is expressed as a linear equation, typically in the form of:

$$\boldsymbol{y = mx + c}$$

- $\boldsymbol{y}$ is the dependent variable.
- $\boldsymbol{x}$ is the independent variables.
- $\boldsymbol{m}$ are the coefficients to be estimated or slope of the line.
- $\boldsymbol{c}$ is the y-intercept

Least Squares Method: Linear regression estimates the coefficients ($\beta\ Value$) by using the method of least squares. It finds the coefficients that minimize the sum of squared differences between the actual values and the values predicted by the linear equation.

Predictions and Inference: Once the coefficients are estimated, you can use the linear equation to make predictions about the dependent variable based on new or existing values of the independent variables. Additionally, linear regression can be used for inference, helping to understand the relationships and significance of the independent variables.

Assumptions: Linear regression makes several assumptions, including linearity, independence of errors, constant variance of errors (homoscedasticity), and normally distributed errors. These assumptions should be checked when using linear regression and can sometimes be addressed with data transformations or other techniques if violated.

Types of Linear Regression:

- **Simple Linear Regression**: Involves one independent variable.
- **Multiple Linear Regression**: Involves multiple independent variables.
- **Polynomial Regression**: Extends linear regression to fit polynomial functions of the predictors.
- **Ridge Regression and Lasso Regression:** Include regularization to prevent overfitting.

Linear regression is widely used in various fields, including economics, finance, biology, social sciences, and machine learning, for tasks such as predicting stock prices, analyzing the impact of factors on sales, and understanding the relationships between variables in scientific studies.

Linear Regression with Python

In AI and machine learning, linear regression is commonly implemented using Python. Python offers a wide range of libraries and tools for linear regression, including NumPy, pandas, and scikit-learn. Here's a step-by-step guide on how to perform linear regression in Python:

1. **Creating NumPy Arrays:** Start by importing the necessary libraries.

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
```

2. **Load Data:** Load your dataset using pandas or any other method suitable for your data source.

```
df = pd.read_csv('homeprices.csv')
df
```

3. **Scatter Plot:** You can visualize by plotting the actual vs. predicted values or any other relevant visualizations.

```
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='red',marker='+')
```

4. **Prepare Data:** Organize your data into the dependent variable (target) and independent variables (features).

```
X = df["area"].values.reshape(-1, 1)
y = df["price"]
```

5. **Create and Train the Model:** Create a linear regression model and fit it to your training data.

```
reg = linear_model.LinearRegression()
reg.fit(X, y)
```

6. **Make Predictions:** Use the trained model to make predictions on the test data.

```
new_y = 3300
reg.predict([[new_y]])
reg.coef_
reg.intercept_
```

7. **Visualize Results (Optional):** You can visualize the results by plotting the actual vs. predicted values or any other relevant visualizations.

```
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='red',marker='+')
plt.plot(df.area, reg.predict(df[['area']]),color='blue')
```

For Multi-Values:

```
import numpy as np
import pandas as pd
from sklearn import linear_model
df = pd.read_csv('homeprices.csv')
df
median_bedrooms= df.bedrooms.median()
median_bedrooms
df.bedrooms = df.bedrooms.fillna(median_bedrooms) # for filling empty
values
df
reg= linear_model.LinearRegression()
reg.fit(df[['area','bedrooms','age']],df.price)
reg.coef_
reg.intercept_
reg.predict([[3000,3,15]])
(112.062*3000+23388.88*3+-3231.71*15)+221323.0018
```

Prediction Function:

A prediction function, often referred to as a predictive model or hypothesis, is a mathematical or computational function that takes input variables and uses them to make predictions or estimates about an unknown or future outcome. In the context of machine learning and statistical modeling, prediction functions are central to the process of making predictions based on data.

Here are the key components of a prediction function:

- **Input Variables (Features):** These are the variables or features that serve as input to the prediction function. The prediction function uses these variables to make predictions. In the context of linear regression, for example, the input variables are the independent variables.
- **Model Parameters (Coefficients or Weights):** Prediction functions typically involve model parameters, such as coefficients or weights, that are learned from training data. These parameters determine how each input variable contributes to the prediction. For instance, in linear regression, the coefficients.

- **Prediction Output:** The prediction function produces an output that is the predicted value of the target variable (dependent variable). In the case of linear regression, the output is a numerical prediction, representing an estimate of the dependent variable based on the values of the independent variables.

Gradient Descent

Gradient descent is an optimization algorithm used to minimize the cost or loss function in machine learning and deep learning models. It's a fundamental algorithm for training models, particularly in cases where the model parameters need to be adjusted to fit the data.

Here's an overview of how gradient descent works:

- **Objective Function (Cost/Loss Function):** In machine learning, you typically have a cost or loss function that measures how well or poorly your model is performing. The goal is to minimize this function. For example, in linear regression, the mean squared error (MSE) can be used as the loss function.
- **Model Parameters (Weights and Biases):** The model you're training has parameters, often represented as weights and biases. These parameters affect the predictions of your model. The goal of training is to find the values of these parameters that minimize the cost function.
- **Gradient Calculation:** Gradient descent computes the gradient (a vector of partial derivatives) of the cost function with respect to the model parameters. This gradient points in the direction of the steepest increase in the cost function. The idea is to move in the opposite direction of the gradient to reach a minimum.

```
#import libraries
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
def gradient_descent(x, y):
    # Initialize the slope (m_curr) and y-intercept (b_curr) to 0
    m_curr = b_curr = 0
    iterations = 1000
    n = len(x)  # Number of data points
    learning_rate = 0.001  # Learning rate for gradient descent

    for i in range(iterations):
        # Calculate the current predicted values based on the current
slope and y-intercept
        y_predicted = m_curr * x + b_curr

        # Calculate the gradients (partial derivatives) of the mean
squared error w.r.t. m and b
        # These gradients will be used to update m_curr and b_curr
        md = -(2/n) * sum(x * (y - y_predicted))
        bd = -(2/n) * sum(y - y_predicted)
```

```
        # Update m_curr and b_curr using the gradients and learning rate
        m_curr = m_curr - learning_rate * md
        b_curr = b_curr - learning_rate * bd

        # Print the updated values of m_curr and b_curr for each
iteration
        print("m {}, b {}, iteration {}".format(m_curr, b_curr, i))

# Sample data points
x = np.array([1, 2, 3, 4, 5])
y = np.array([5, 7, 9, 11, 13])

# Call the gradient_descent function to fit a line to the data
gradient_descent(x, y)
```

Gradient Descent for Cost:

```
#import libraries
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
def gradient_descent(x, y):
    # Initialize the slope (m_curr) and y-intercept (b_curr) to 0
    m_curr = b_curr = 0
    iterations = 1000
    n = len(x)  # Number of data points
    learning_rate = 0.01  # Learning rate for gradient descent

    for i in range(iterations):
        # Calculate the current predicted values based on the current
slope and y-intercept
        y_predicted = m_curr * x + b_curr

        # Calculate the mean squared error (cost) as a measure of how
well the model fits the data
        cost = (1/n) * sum([val**2 for val in (y - y_predicted)])

        # Calculate the gradients (partial derivatives) of the cost
w.r.t. m and b
        # These gradients will be used to update m_curr and b_curr
        md = -(2/n) * sum(x * (y - y_predicted))
```

```
        bd = -(2/n) * sum(y - y_predicted)

        # Update m_curr and b_curr using the gradients and learning rate
        m_curr = m_curr - learning_rate * md
        b_curr = b_curr - learning_rate * bd

        # Print the updated values of m_curr, b_curr, cost, and the
iteration number
        print("m {}, b {}, cost {} iteration {}".format(m_curr, b_curr,
cost, i))

# Sample data points
x = np.array([1, 2, 3, 4, 5])
y = np.array([5, 7, 9, 11, 13])

# Call the gradient_descent function to fit a line to the data
gradient_descent(x, y)
```

To Save Model:

1. **Pickle:** Pickle is a Python module that provides a way to serialize and deserialize (serialize means convert to a byte stream, and deserialize means convert from a byte stream) Python objects. Serialization is the process of converting complex data structures, such as lists, dictionaries, and custom objects, into a format that can be easily saved to a file or transmitted over a network. Deserialization is the reverse process of reconstructing the original Python objects from the serialized data.

```
# Saving model by Pickle
# Import Library
import pickle
#to save model
with open('model_pickle','wb') as f:
  pickle.dump(model,f)
#using saved model
with open('model_pickle','rb') as f:
  mp = pickle.load(f)
mp.predict([[5000]])
```

2. **Joblib:** Joblib is a Python library that provides tools for lightweight pipelining in Python, particularly for efficiently saving and loading Python objects (including NumPy arrays and machine learning models) to and from disk. It is especially well-suited for tasks involving scientific computing and machine learning, where you often need to serialize and deserialize large data structures and models.

```
# Saving model by Joblib
# import library
import joblib
```

```
#to save model
joblib.dump(model,'model_joblib')
#using saved model
mj = joblib.load('model_joblib')
mj.predict([[5000]])
```

Lab Task

1. Perform a simple linear regression analysis using the sklearn library to predict a target variable based on a single feature. Dataset: Use the "Boston Housing Prices" dataset available in sklearn.datasets. The task is to predict the median value of owner-occupied homes (target) based on a single feature like the number of rooms.
2. Extend the previous task to perform a multiple linear regression analysis using sklearn with multiple features.Dataset: Use the "Boston Housing Prices" dataset from sklearn.datasets. The task is to predict the median value of owner-occupied homes (target) using multiple features such as the number of rooms, crime rate, etc.

Home Task

1. Perform a simple linear regression analysis using the sklearn library to predict a target variable based on a single feature. Dataset: Use the "Titanic" dataset availableon Github.
2. Extend the previous task to perform a multiple linear regression analysis using sklearn with multiple features. Dataset: Use the "Titanic" dataset availableon Github.
3. Implement the gradient descent algorithm from scratch in Python.Dataset: Generate a synthetic dataset of your choice with multiple features and a target variable. Note : Students should implement the gradient descent algorithm, perform feature scaling (if needed), and use gradient descent to optimize a linear regression model. They should plot the cost function over iterations and analyze the learning process.