Applied Machine Learning

Assignment I

Submitted to:
Dr. Tanupriya Choudhary

Submitted by :
Rajkumar Pandey
SAP : 590011165

UPES
UNIVERSITY OF TOMORROW

Q-1: what type of ML algorithm used in a dataset how to find it out ?

Ans: Identifying the type of machine learning (ML) algorithm used for a given dataset requires understanding the dataset's characteristics, the problem it aims to solve, and the nature of its target variable.

1.Understand the Problem:

i. **Supervised Learning**: Classification (e.g., spam detection) or Regression (e.g., house prices).

ii. **Unsupervised Learning**: Clustering (e.g., customer segmentation) or Dimensionality Reduction (e.g., PCA).

iii. **Reinforcement Learning**: Learn by interaction (e.g., game playing).

2.Analyze the Dataset:

i. **Structured Data**: Use Decision Trees, Random Forests, or Linear Regression.

ii. **Unstructured Data**: Use CNNs (images), RNNs (text), or Transformers.

iii. **Small Dataset**: Use simpler models (e.g., Logistic Regression).

iv. **Large Dataset**: Use complex models (e.g., Deep Learning).

v. **Numerical Features**: Use Linear Regression, SVM, or Neural Networks.

vi. **Categorical Features**: Use Decision Trees, Random Forests, or XGBoost.

vii. **Labeled Data**: Use supervised learning.

viii. **Unlabeled Data**: Use unsupervised learning.

3.Choose the Algorithm:

i. **Classification**: Logistic Regression, Decision Trees, Random Forests, SVM, Neural Networks.

ii. **Regression**: Linear Regression, Ridge Regression, Random Forests, XGBoost.

iii. **Clustering**: K-Means, DBSCAN, Gaussian Mixture Models.

iv. **Dimensionality Reduction**: PCA, t-SNE, Autoencoders.

v. **Anomaly Detection**: Isolation Forest, One-Class SVM.

vi. **Recommendation Systems**: Collaborative Filtering, Neural Networks.

4.Evaluate and Iterate:

i. Split data into training and testing sets.

ii. Train the model and evaluate performance (e.g., accuracy, RMSE).

iii. Refine the model by tuning hyperparameters or trying different algorithms.

Q-2: What is the difference between hinge loss and triplet loss explain with an example?

nature	Triplet Loss	Hinge Loss
Purpose	Used in **metric learning** to ensure that similar samples (anchor-positive) are closer together while dissimilar ones (anchor-negative) are farther apart.	Used in **classification tasks** (mainly for SVMs) to maximize the margin between positive and negative classes.
Formula	**max(d(A,P)−d(A,N)+margin,0)** where A is the anchor, P is the positive sample, and N is the negative sample.	**max(0,1−yf(x))** where y is the true label (+1 or -1), and f(x) is the predicted score.
Key Concept	Minimizes the distance between similar points while pushing apart dissimilar ones.	Ensures a minimum margin between classes in classification.
Usage	Used in **face recognition**, **metric learning**, and **embedding learning**.	Common in **Support Vector Machines (SVMs)** for classification.
Hyperparameter	Requires a **margin** hyperparameter to define the minimum separation between positive and negative pairs.	Uses a **margin (1 by default)** in SVMs to define the separation boundary.

UPES
UNIVERSITY OF TOMORROW

Hinge loss:

Hinge Loss = max(0,1 − y · y^)

- Y : True label (+1 or −1).
- *y*^: Raw Predicted score (not probability).

Example:

True label (*y*): (+1 or –1)

Predicted score (*y*^): 00.8

Hinge Loss:

Hinge Loss = max(0,1−(1·0.8))=max(0,0.2)=0.2

Triplet loss:

Triplet Loss=max(0,*d*(*a*,*p*) − *d*(*a*,*n*) + margin)

p: Positive example (same class as anchor).

n: Negative example (different class from anchor).

d(·): Distance metric (e.g., Euclidean distance).

a: Anchor example

Example:

Anchor (a): [1.0,2.0]

Positive (p): [1.1,2.2]

Negative (n): [3.0,4.0]

Margin: 0.5

$$d_{\text{anchor-positive}} = \sqrt{(1.0-1.1)^2 + (2.0-2.2)^2} \approx 0.2236$$

$$d_{\text{anchor-negative}} = \sqrt{(1.0-3.0)^2 + (2.0-4.0)^2} \approx 2.828$$

Calculate loss:

$$\text{Loss} = \max(0, 0.2236 - 2.828 + 0.5) = \max(0, -2.104) = 0$$

UPES
UNIVERSITY OF TOMORROW

Q-3:what do you mean by traianing testing validation of data showcase with help of python programming example?

Ans: **Training Data:**

- This is the actual dataset from which a model trains .i.e. the model sees and learns from this data to predict the outcome or to make the right decisions.
- Most of the training data is collected from several resources and then preprocessed and organized to provide proper performance of the model.
- Type of training data hugely determines the ability of the model to generalize .i.e. the better the quality and diversity of training data, the better will be the performance of the model. This data is more than 60% of the total data available for the project.

Testing Set:

- This dataset is independent of the training set but has a somewhat similar type of probability distribution of classes and is used as a benchmark to evaluate the model, used only after the training of the model is complete.
- Testing set is usually a properly organized dataset having all kinds of data for scenarios that the model would probably be facing when used in the real world If the accuracy of the model on training data is greater than that on testing data then the model is said to have overfitting.
- This data is approximately 20-25% of the total data available for the project.

Validation set:

- The validation set is used to fine-tune the hyperparameters of the model and is considered a part of the training of the model.
- Validation dataset can be utilized for regression as well by interrupting training of model when loss of validation dataset becomes greater than loss of training dataset .i.e. reducing bias and variance.
- This data is approximately 10-15% of the total data available for the project but this can change depending upon the number of hyperparameters .i.e. if model has quite many hyperparameters then using large validation set will give better results.
- Now, whenever the accuracy of model on validation data is greater than that on training data then the model is said.

```python
# Importing numpy & scikit-learn
import numpy as np
from sklearn.model_selection import train_test_split
# Making a array for x ranging from 0-23 then reshaping it
# to form a matrix of shape 8x3
x = np.arange(24).reshape((8,3))
# y is just a list of 0-7 number representing
# target variable
y = range(8)
# Splitting dataset in 80-20 fashion .i.e.
# Training set is 80% of total data
# Combined set of testing & validation is
# 20% of total data
x_train, x_Combine, y_train, y_Combine = train_test_split(x,y,
                                            train_size=0.8,
                                          random_state=42)
# Splitting combined dataset in 50-50 fashion .i.e.
# Testing set is 50% of combined dataset
# Validation set is 50% of combined dataset
x_val, x_test, y_val, y_test = train_test_split(x_Combine,
                                                y_Combine,
                                                test_size=0.5,
                                                random_state=42)

# Training set
print("Training set x: ",x_train)
print("Training set y: ",y_train)
print(" ")
# Testing set
print("Testing set x: ",x_test)
print("Testing set y: ",y_test)
print(" ")
# Validation set
print("Validation set x: ",x_val)
print("Validation set y: ",y_val)
```

UPES
UNIVERSITY OF TOMORROW

```
Training set x:  [[ 0  1  2]
 [21 22 23]
 [ 6  7  8]
 [12 13 14]
 [ 9 10 11]
 [18 19 20]]
Training set y:  [0, 7, 2, 4, 3, 6]

Testing set x:  [[15 16 17]]
Testing set y:  [5]

Validation set x:  [[3 4 5]]
Validation set y:  [1]
```

Q-4 what is the use of training & testing of data and in training and testing how random operator can be used?

Ans: In machine learning, the process of training and testing data is essential for building and evaluating models.

1.Training Data:

1. **Purpose**: Training data is used to teach the model to identify patterns, relationships, or features within the data. The model learns by adjusting its parameters (e.g., weights in a neural network) to reduce errors.
2. **Use**: The model "learns" from the training data to make predictions or decisions.

2.Testing Data:

1. **Purpose**: Testing data is used to evaluate the model's performance on unseen data. It helps determine how well the model generalizes to new inputs.
2. **Use**: The model's predictions on the testing data are compared to the actual values to measure accuracy, precision, recall, or other performance metrics.

Why Split Data into Training and Testing Sets?

•**Prevent Overfitting**: If a model is trained and evaluated on the same data, it may perform well on that data but fail to generalize to new, unseen data. Splitting the data ensures the model is tested on data it has never encountered.

Role of random operator in training and testing:

A **random operator** is often used in the process of splitting data into training and testing sets.

Random Splitting: Divides data into training/testing sets fairly.

Reproducibility: Fixed seed (e.g., random_state=42) ensures consistent results.

Prevents Bias: Avoids skewed splits from ordered data.

Cross-Validation: Randomly creates subsets for robust evaluation.

Parameter Initialization: Randomly sets model weights to avoid poor optima.

Fair Testing: Ensures unbiased evaluation on unseen data.

Code:

```
from sklearn.model_selection import train_test_split
import numpy as np
# Example dataset
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([0, 1, 0, 1])
# Randomly split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
print("Training data:", X_train)
print("Testing data:", X_test)
```

```
Training data: [[7 8]
 [1 2]
 [5 6]]
Testing data: [[3 4]]
```

UPES
UNIVERSITY OF TOMORROW

Q-5 What is the difference between sample and population variance share with the help of equation and example.

Aspect	Sample Variance (s^2)	Population Variance (σ^2)
Definition	Measures the variance of a **sample** (subset of data) from a population.	Measures the variance of an entire **population** (all data points).
Formula	$\frac{\sum(X_i-\bar{X})^2}{n-1}$	$\frac{\sum(X_i-\mu)^2}{N}$
Mean Used	Sample mean ($\bar{x}$)	Population mean (μ)
Denominator	n − 1 (degrees of freedom)	*N* (total number of data points)
Purpose	Estimates the population variance from a sample	Measures the true variance of the population
Bias Adjustment	Dividing by n − 1 corrects the bias in estimation	No bias adjustment needed, uses entire population data
Usage	When only a subset of the population is available	When the entire population data is available
Data Set	Sample data (a subset of the population)	Population data (all data points in the population)

UPES
UNIVERSITY OF TOMORROW

Example

Dataset

Consider the following dataset representing the ages of 5 people in a small town (population):

Ages={25,30,35,40,45}

Population Variance:

1.Calculate the population mean (μ):

$\mu=(25+30+35+40+45)/5 = 35$

2.Calculate squared deviations from the mean:

$(25-35)^2=100$, $(30-35)^2=25$, $(35-35)^2=0$, $(40-35)^2=25$, $(45-35)^2=100$

3.Sum the squared deviations:

$100+25+0+25+100=250$

4.Divide by N (population size):

$\sigma2=250/5=50$

Sample Variance:

Now, assume we take a sample of 3 ages from the population:

1.Sample Ages={25,30,35}

$\bar{x}=(25+30+35)/3=30$

2.Calculate squared deviations from the mean:

$(25-30)^2=25$,$(30-30)^2=0$,$(35-30)^2=25$

3.Sum the squared deviations:

$25+0+25=50$

4.Divide by n−1 (sample size minus 1):

$S2=50/(3-1)=25$

UPES
UNIVERSITY OF TOMORROW

Q-6:in the loss function what is the meaning of one hot encoding and integer encoding?

Ans:

One-Hot Encoding:

One-hot encoding converts categories into **binary vectors**, where each category is represented as a vector of 0s and 1s

How it is Used in Loss Functions?

- **With One-Hot Encoding**:
  - The model outputs probability distributions using **softmax activation**.
  - The **cross-entropy loss function** compares the predicted probabilities with the actual one-hot encoded labels.

Example: Consider a dataset with three classes: Car, Plane, Ship

Category	One-Hot Encoding
Car	[1, 0, 0]
Plane	[0, 1, 0]
Ship	[0, 0, 1]

Integer Encoding

Integer encoding assigns a unique integer to each category.

How it is Used in Loss Functions?

Typically used with **sparse categorical cross-entropy**, which converts integer labels to one-hot internally.

Example: Consider a dataset with three classes: Car, Plane, Ship

Category	Integer Encoding
Car	0
Plane	1
Ship	2

UPES
UNIVERSITY OF TOMORROW

Q-7 In ml what do mean by overfitting of a model and how we are dealing with overfitting show case with the help of the python programming

ANS: **Overfitting:** Overfitting happens when a machine learning model **learns the noise** in the training data instead of the actual pattern.

This leads to:

- High accuracy on the **training data**
- Poor performance on **new/unseen data**

Example Scenario:

A student memorizes answers instead of understanding concepts. They score well in practice exams but fail real tests with new questions.

How to deal with overfitting:

1.Train with More Data – Helps the model learn general patterns instead of memorizing.

2. Cross-Validation – Splitting the dataset into multiple parts (e.g., K-Fold Cross-Validation).

3. Regularization – Techniques like **L1 (Lasso) and L2 (Ridge) Regularization** add penalty terms to prevent overfitting.

4. Dropout (For Neural Networks) – Randomly deactivates some neurons during training.

5. Pruning (For Decision Trees) – Removing less important branches.

6. Early Stopping – Stops training when validation loss starts increasing

Code:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
# Generate a simple regression dataset
np.random.seed(42)
X = np.random.rand(100, 1)  # 100 samples, 1 feature
y = 3 * X.squeeze() + np.random.randn(100) * 0.1  # y = 3x + noise
# Split into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
# Get training and test scores
train_score = r2_score(y_train, model.predict(X_train))
test_score = r2_score(y_test, model.predict(X_test))
print(f"Training R² Score: {train_score:.4f}")
print(f"Test R² Score: {test_score:.4f}")
# Detect overfitting
if train_score > 0.9 and (train_score - test_score) > 0.2:
    print("⚠️ Model is overfitting!")
else:
    print("✅ Model is generalizing well.")
```

```
Training R² Score: 0.9887
Test R² Score: 0.9924
✅ Model is generalizing well.
```

UPES
UNIVERSITY OF TOMORROW

```python
from sklearn.linear_model import Ridge
# Train Ridge Regression with L2 Regularization
ridge_model = Ridge(alpha=1.0)  # Regularization strength
ridge_model.fit(X_train, y_train)
# Recalculate accuracy
ridge_train_score = r2_score(y_train, ridge_model.predict(X_train))
ridge_test_score = r2_score(y_test, ridge_model.predict(X_test))
print("\nAfter Fixing Overfitting:")
print(f"Training R² Score: {ridge_train_score:.4f}")
print(f"Test R² Score: {ridge_test_score:.4f}")
print("✅ Overfitting reduced!")
```

```
After Fixing Overfitting:
Training R² Score: 0.9724
Test R² Score: 0.9779
✅ Overfitting reduced!
```

UPES
UNIVERSITY OF TOMORROW

Q-8 How to handle noisy data and also what do you mean by Imbalance data showcase with example in python programming. What type of measurement we will be taking to handle the data set.

Ans: **1.Noisy Data:** Noisy data contains **errors, outliers, or irrelevant information** that can mislead machine learning models.

How to Handle Noisy Data?

- **Smoothing Techniques** – Use moving averages, Gaussian filters, or regression models.
- **Outlier Detection & Removal** – Methods like Z-score, IQR, or Isolation Forest.
- **Data Cleaning** – Fix missing values using **mean imputation** or **interpolation**.

2.Imbalanced Data:

Imbalanced data occurs when one class significantly outnumbers the other(s) in a classification problem. This can lead to biased models that favor the majority class.

How to Handle Imbalanced Data:

- **Resampling**:
  - **Oversampling**: Increase the number of minority class samples (e.g., using SMOTE).
  - **Undersampling**: Reduce the number of majority class samples.
- **Class Weights**:
  - Assign higher weights to the minority class during model training.
- **Algorithm Selection**:
  - Use algorithms that handle imbalanced data well (e.g., Random Forests, Gradient Boosting).
- **Evaluation Metrics**:
  - Use metrics like **F1-score**, **Precision-Recall Curve**, or **AUC-ROC** instead of accuracy.

UPES
UNIVERSITY OF TOMORROW

Code:

Noisy data:

```
import numpy as np
import pandas as pd
from scipy import stats
# Create a noisy dataset
data = np.array([10, 12, 11, 15, 100, 13, 14, 12])  # 100 is an outlier
# Detect and remove outliers using Z-score
z_scores = np.abs(stats.zscore(data))
filtered_data = data[(z_scores < 2)]  # Keep data within 2 standard deviations
print("Original Data:", data)
print("Filtered Data:", filtered_data)
```

```
Original Data: [ 10  12  11  15 100  13  14  12]
Filtered Data: [10 12 11 15 13 14 12]
```

Imbalance data:

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE
# Create an imbalanced dataset
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, weights=[0.99, 0.01], random_state=42)
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Apply SMOTE to oversample the minority class
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
# Train a model
model = RandomForestClassifier(random_state=42)
model.fit(X_resampled, y_resampled)
# Evaluate
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       296
           1       0.00      0.00      0.00         4

    accuracy                           0.98       300
   macro avg       0.49      0.49      0.49       300
weighted avg       0.97      0.98      0.98       300
```

UPES
UNIVERSITY OF TOMORROW

Q-9: Share with the example of k-fold cross validation technique. you may take k with any number like 5,10,20,30.

ANS: K-Fold Cross-Validation is a technique used to evaluate the performance of a machine learning model. It divides the dataset into **k subsets (folds)**, trains the model on **k-1 folds**, and validates it on the remaining fold. This process is repeated **k times**, with each fold used exactly once as the validation set. The final performance is the average of the results from all folds.

Why use it?

- Prevents **overfitting**
- Ensures **better model evaluation**
- Useful when **data is limited**

steps of K-Fold Cross-Validation

1.Split the dataset into **k equal parts (folds)**.

2.For each fold:

1. Train the model on **k-1 folds**.
2. Validate the model on the **remaining fold**.

3.Calculate the average performance across all folds.

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
# Load sample dataset
data = load_iris()
X, y = data.data, data.target
# Define K-Fold Cross-Validation
k = 5  # Number of folds
kf = KFold(n_splits=k, shuffle=True, random_state=42)
# Initialize model
model = RandomForestClassifier()
# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf, scoring='accuracy')

# Print Results
print(f'Accuracy for each fold: {scores}')
print(f'Mean Accuracy: {np.mean(scores):.4f}')
```

```
Accuracy for each fold: [1.         0.96666667 0.93333333 0.93333333 0.93333333]
Mean Accuracy: 0.9533
```

UPES
UNIVERSITY OF TOMORROW

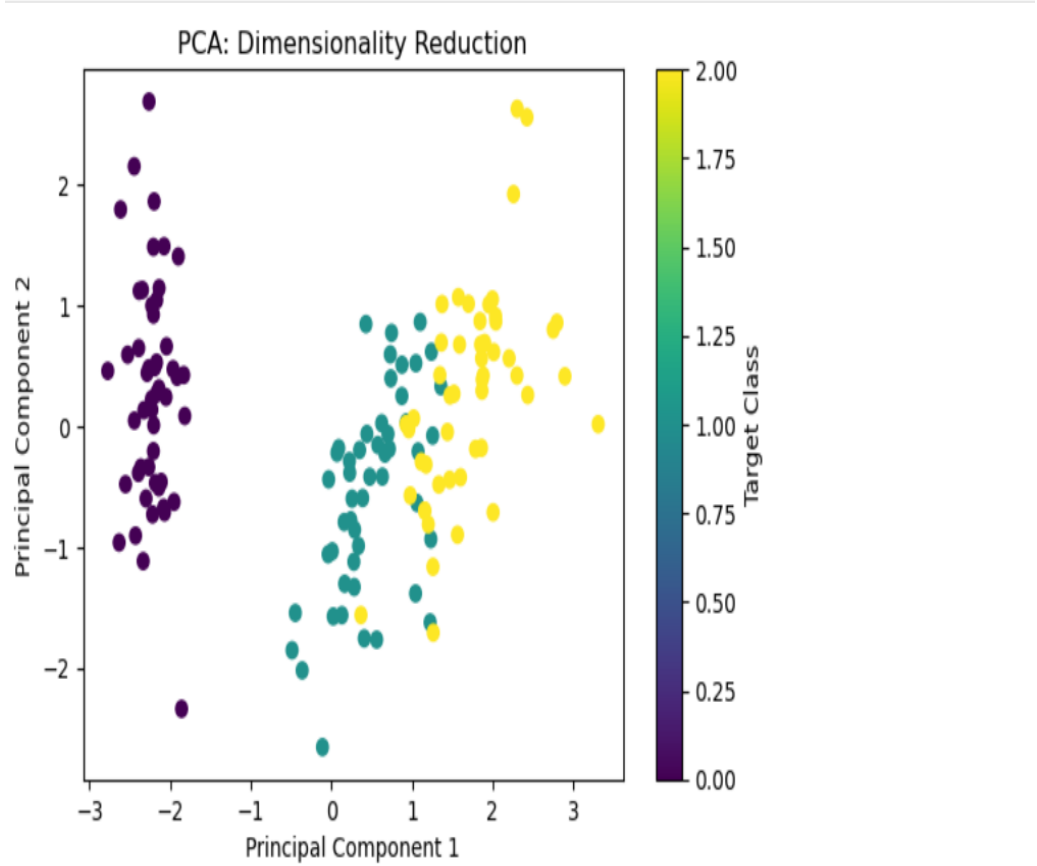
Q-10:What is the difference between dimensionality reduction and dimensionality addition? Share a case study using the PCA algorithm.
Ans:

Aspect	Dimensionality Reduction	Dimensionality Addition
Definition	Reduces the number of features (dimensions) in the dataset while retaining important information.	Increases the number of features by creating new ones, often through feature engineering.
Purpose	Simplifies the dataset, reduces noise, and improves computational efficiency.	Enhances the dataset by adding meaningful features to improve model performance.
Techniques	PCA, t-SNE, LDA, Autoencoders, etc.	Polynomial features, interaction terms, domain-specific feature creation, etc.
Use Case	High-dimensional data (e.g., images, text) where reducing dimensions is necessary.	Low-dimensional data where additional features can provide more insights.
Impact on Model	Speeds up training, reduces overfitting, and improves interpretability.	Can improve model accuracy by capturing complex relationships in the data.

Case Study: Using PCA for Dimensionality Reduction

Scenario: We have a dataset with **10 features**, and we want to reduce them to **2 principal components** while maintaining most of the variance.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
# Load dataset
data = load_iris()
X = data.data  # Features
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Apply PCA (reduce to 2 dimensions)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
# Convert to DataFrame
df_pca = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
# Plot PCA result
plt.scatter(df_pca['PC1'], df_pca['PC2'], c=data.target, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA: Dimensionality Reduction')
plt.colorbar(label='Target Class')
plt.show()
```



UPES
UNIVERSITY OF TOMORROW