Practical-1

AIM: Study Of Machine learning basics

1. What is Machine learning

**SOLUTION:**

Machine learning is a part of artificial intelligence that allows computers to learn from data and improve their performance on a task without being explicitly programmed. It's like teaching machines to learn and make decisions based on experience.

2. Steps in collection of data

**SOLUTION**:

1. **Define Objectives:** Clearly outline the goals of data collection. What insights or information are you seeking?
2. **Plan and Design:** Determine the type of data needed and how it will be collected. Design surveys, experiments, or other methods accordingly.
3. **Select Data Sources:** Identify where to find the required data. This could involve surveys, existing databases, sensors, or other sources.
4. **Data Collection:** Execute the planned methods to gather data. This might involve surveys, observations, interviews, or automated sensors, depending on the nature of the study.
5. **Data Validation:** Ensure the collected data is accurate and reliable. This step involves checking for errors and inconsistencies.
6. **Data Entry:** If applicable, enter the data into a computer system for analysis. This step is crucial for manual data collection methods.
7. **Data Cleaning:** Identify and rectify any errors or inconsistencies in the collected data. This ensures the data is ready for analysis.
8. **Data Storage:** Organize and store the data securely. This step is critical for maintaining data integrity and privacy.

3. Steps in importing the data in python (Through: csv, json, and other data formats)

**SOLUTION**:

import csv # For CSV files

import json # For JSON files

import pandas as pd # For various data formats

### Import CSV Data:

with open('your\_file.csv', 'r') as file:

csv\_reader = csv.reader(file)

data = [row for row in csv\_reader]

### Import JSON Data:

with open('your\_file.json', 'r') as file:

data = json.load(file)

# For CSV, Excel, JSON, and more

data\_csv = pd.read\_csv('your\_file.csv')

data\_excel = pd.read\_excel('your\_file.xlsx')

data\_json = pd.read\_json('your\_file.json')

# Display the first few rows of a pandas DataFrame

print(data\_csv.head())

print(data\_excel.head())

print(data\_json.head())

4. Preprocessing

1. Remove Outliers

**SOLUTION**:

import pandas as pd

from scipy import stats

# Assuming 'data' is a pandas DataFrame

z\_scores = stats.zscore(data)

filtered\_data = data[(z\_scores < 3).all(axis=1)] # Keep data points within 3 standard deviations

2. Normalize Datasets, Data encoding

**SOLUTION**:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

normalized\_data = scaler.fit\_transform(data)

DATA ENCODING:  
# Assuming 'data' has categorical columns to be encoded

encoded\_data = pd.get\_dummies(data)

# Alternatively, you can use Label Encoding for ordinal categorical data

# from sklearn.preprocessing import LabelEncoder

# label\_encoder = LabelEncoder()

# encoded\_data['categorical\_column'] = label\_encoder.fit\_transform(data['categorical\_column'])

3. Handling Missing Data

**SOLUTION**:  
# Drop rows with missing values

data\_without\_missing = data.dropna()

# Fill missing values with the mean of the column

data\_filled\_mean = data.fillna(data.mean())

5. Machine Models

1. Types of machine learning models – Supervised learning, Unsupervised learning, reinforcement learning.

SOLUTINON:

a. Supervised Learning:

Definition: In supervised learning, the model is trained on a labeled dataset, where the input data is paired with corresponding target labels.

Example: Classification and Regression.

b. Unsupervised Learning:

Definition: Unsupervised learning involves training models on unlabeled data, and the algorithm tries to learn the patterns and structure within the data.

Example: Clustering and Dimensionality Reduction.

c. Reinforcement Learning:

Definition: Reinforcement learning involves an agent learning how to behave in an environment by performing actions and receiving rewards or penalties in return.

Example: Training a computer program to play a game.

2. Parameters of machine learning model (Learning rate, regularization, etc.)

**SOLUTION**:

### Learning Rate:

The learning rate is crucial, influencing how quickly a model converges and its overall performance. If set too high, the model might overshoot the optimal solution; if too low, convergence may be slow.

### Regularization:

Regularization combats overfitting by adding a penalty term to the loss function. L1 regularization (Lasso) and L2 regularization (Ridge) are common methods.

### Hyperparameters:

These are pre-training external settings, not learned from data. Examples include the number of hidden layers in a neural network or the number of trees in a random forest.

### Activation Function:

Activation functions like ReLU, Sigmoid, and Tanh introduce non-linearity, influencing the model's capacity to learn complex patterns.

### Loss Function:

The loss function measures the disparity between predicted and actual values, guiding model training. For regression, Mean Squared Error (MSE); for classification, Cross-Entropy is often used.

6. Test-train data split: using constant ration, k-fold cross validation

**SOLUTION**:  
a. Using Constant Ratio:

Divide the dataset into two parts, typically 70-30 or 80-20 for training and testing, respectively. The larger portion is used for training the model, and the smaller one is reserved for evaluating its performance. This simple approach provides a quick assessment but may lead to variability in results depending on the random split.

b. K-Fold Cross-Validation:

1. \*\*Divide Data into K Folds:\*\*

- Split the dataset into K equally-sized folds.

2. \*\*Iterate Through Folds:\*\*

- For each fold, designate it as the test set and the remaining K-1 folds as the training set.

3. \*\*Train and Evaluate:\*\*

- Train the model on the training set and evaluate its performance on the test set.

4. \*\*Average Results:\*\*

- Repeat this process K times, using a different fold as the test set in each iteration.

- Average the performance metrics to get a more reliable estimate of the model's performance.

7. Output Inference

**SOLUTION**:  
1. \*\*Prediction Interpretation:\*\*

- Examine the model's predictions in the context of the problem. Understand the meaning and implications of the predicted values.

2. \*\*Thresholding (if applicable):\*\*

- For classification tasks, apply a threshold to convert raw model outputs into class predictions. This step is essential when dealing with probability scores.

3. \*\*Evaluate Confidence:\*\*

- Assess the model's confidence in its predictions. Some models provide uncertainty estimates or confidence intervals.

4. \*\*Error Analysis:\*\*

- Investigate prediction errors. Identify patterns or specific cases where the model struggles, helping improve model performance.

5. \*\*Business/Application Context:\*\*

- Consider the broader business or application context. How do the model's predictions impact decision-making or actions?

6. \*\*Communication of Results:\*\*

- Clearly communicate the model's outputs and any associated uncertainties to relevant stakeholders. Effective communication ensures informed decision-making.

7. \*\*Iterative Improvement:\*\*

- If applicable, use feedback and new data to iteratively improve the model. Continuous learning and refinement contribute to better performance over time.

8. \*\*Decision Making:\*\*

- Based on the model's predictions, make informed decisions or take appropriate actions in line with the problem's objectives.

8. Validation: different metrics – Confusion Matrix, Precision, Recall, F1-score

**SOLUTION**:  
1. Confusion Matrix:

- A table that summarizes the performance of a classification model.

- \*\*Components:\*\*

- \*\*True Positive (TP):\*\* Correctly predicted positive instances.

- \*\*True Negative (TN):\*\* Correctly predicted negative instances.

- \*\*False Positive (FP):\*\* Incorrectly predicted as positive.

- \*\*False Negative (FN):\*\* Incorrectly predicted as negative.

2. Precision:

- Precision is the ratio of correctly predicted positive observations to the total predicted positives.

- \*\*Formula:\*\* Precision = TP / (TP + FP)

- A high precision indicates a low false positive rate.

3. Recall (Sensitivity):

- Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

- \*\*Formula:\*\* Recall = TP / (TP + FN)

- A high recall indicates a low false negative rate.

4. F1-Score:

- F1-Score is the harmonic mean of precision and recall, providing a balanced evaluation metric.

- \*\*Formula:\*\* F1-Score = 2 \* (Precision \* Recall) / (Precision + Recall)

- Particularly useful when there is an uneven class distribution.

Application:

- Use these metrics to comprehensively evaluate a classification model's performance, especially in scenarios where certain errors (false positives or false negatives) carry more significance.

- Consider the specific requirements of your problem to choose the most appropriate metric for model validation.