Artificial Intelligence for Cybersecurity
Experiment 3
Machine learning – Classification

Machine learning, deep learning and artificial intelligence

Machine learning (ML), deep learning (DL) and artificial intelligence (AI) are related concepts in the field of computer science and artificial intelligence.

ML is a subfield of AI that focuses on the development of algorithms and statistical models that enable computers to improve their performance on a specific task without being explicitly programmed.

DL is a type of machine learning that uses neural networks with multiple hidden layers to learn and make predictions based on large amounts of data.

AI is the broader concept that encompasses ML and DL, as well as other advanced technologies such as computer vision, natural language processing, and robotics. AI refers to the development of intelligent machines that can perform tasks that normally require human intelligence, such as recognizing speech, playing chess, and solving problems.

In summary, ML and DL are subfields of AI, and ML is a broad category that encompasses DL and other types of machine learning algorithms.

ML algorithms

ML algorithms can be broadly classified into the following categories:

1. **Supervised Learning**: Algorithms that learn from labelled data, where the desired output is already known. The labelled data contains both the input (features) and the output (target) variables. The goal of supervised learning is to learn a mapping function that can predict the output variable for new, unseen input data.

   Classification is a type of supervised learning algorithm where the goal is to predict a categorical or discrete target variable. For example, classifying emails as spam or not spam, or classifying images of animals as cats or dogs.

   Regression is another type of supervised learning algorithm where the goal is to predict a continuous or numerical target variable. For example, predicting the price of a house based on its features such as location, size, number of bedrooms, etc.

Examples of supervised learning algorithms include –

   a. **linear regression**
   b. **logistic regression**
   c. **decision trees**
   d. **random forests**
   e. **support vector machines (SVM).**

2. Unsupervised learning: Algorithms that learn from unlabelled data, where the desired output is unknown. Examples of unsupervised learning algorithms include k-means clustering, hierarchical clustering, and dimensionality reduction techniques like principal component analysis (PCA).
3. Reinforcement learning
4. Semi-supervised Learning
5. Transfer Learning

The choice of which algorithm to use depends on the specific problem and the type of data available.

Example Implementations of ML

Decision trees

Decision trees are a type of machine learning algorithm that is used for both regression and classification problems. They are called "trees" because the prediction process is represented in a tree-like structure, where each internal node represents a decision based on a certain feature, and each leaf node represents a final prediction.

The decision tree algorithm works by splitting the data into smaller subsets based on the values of the features. This process is repeated recursively until a stopping condition is reached, such as a minimum number of samples in a leaf node or a maximum tree depth. The final prediction is made by starting at the root of the tree and following the path that corresponds to the feature values of the data point to be predicted.

A step-by-step tutorial to perform decision tree-based classification on an example dataset from scikit-learn:

Step 1: Load necessary libraries and the example dataset

```
# Importing libraries
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Loading example dataset
```

```
from sklearn.datasets import load_iris
iris = load_iris()
```

In the code above, we have imported the necessary libraries such as numpy, pandas, and scikit-learn. We have also loaded the Iris dataset from scikit-learn. The Iris dataset is a well-known dataset in the machine learning community and is used as an example for classification problems. The dataset contains 150 samples of iris flowers, and the goal is to predict the species of the iris based on four features: sepal length, sepal width, petal length, and petal width.

Step 2: Split the data into training and testing sets

```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2)
```

In this step, we have used the train_test_split function from scikit-learn to split the data into training and testing sets. The train_test_split function takes the features iris.data and target iris.target as input, and the test_size parameter is set to 0.2, which means that 20% of the data will be used for testing and 80% of the data will be used for training.

Step 3: Fit the decision tree model to the training data

```
# Fitting the decision tree model to the training data
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

In this step, we have created a DecisionTreeClassifier object and fit it to the training data using the fit method. The decision tree algorithm will use the training data to build the tree structure and make predictions.

Step 4: Make predictions on the test data

```
# Making predictions on the test data
y_pred = clf.predict(X_test)
```

In this step, we have used the predict method to make predictions on the test data. The predict method takes the test data as input and returns the predicted target values.

Step 5: Evaluate the performance of the model by calculating the accuracy score

```
# Evaluating the performance of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

In this step, we have used the accuracy_score function from scikit-learn to calculate the accuracy of the model. The accuracy_score function takes the actual target values y_test and the predicted target values y_pred as input and returns the accuracy score. The accuracy score is a commonly used metric for classification problems and represents the proportion of correct predictions made by the model.

Random forest

Random forest is a popular machine learning algorithm used for both regression and classification problems. It is a type of ensemble learning method, where multiple decision trees are combined to make a prediction. The idea behind this approach is to aggregate the results of many trees to get a more accurate prediction.

Each tree in the forest is trained on a random subset of the data and the final prediction is made by aggregating the results of all the trees. This process helps to reduce the risk of overfitting, which occurs when a model is too closely fit to the training data and performs poorly on new, unseen data.

Random forest is a flexible and easy-to-use algorithm that can be applied to a wide range of problems. It is widely used in industries such as finance, healthcare, and retail, where large amounts of data are collected and analysed. Despite its simplicity, Random Forest often provides results that are comparable or even better than more complex algorithms.

A step-by-step tutorial to perform Random forest classification on an example dataset from scikit-learn:

Step 1: Import Required Libraries

```
# Import the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

In this step, we've imported the required libraries:

- numpy for numerical computing
- pandas for data manipulation and analysis
- matplotlib.pyplot for plotting the results
- train_test_split from sklearn.model_selection for splitting the data into training and testing sets
- RandomForestClassifier from sklearn.ensemble for training the Random Forest Classifier model

- confusion_matrix and classification_report from sklearn.metrics for evaluating the model performance

Step 2: Load the Dataset

```
# Load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df["target"] = iris.target
```

In this step, we've loaded the iris dataset from scikit-learn and convert it into a pandas DataFrame for easier manipulation and analysis. The iris dataset contains 4 features and 1 target variable.

Step 3: Data Preprocessing

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(iris_df.iloc[:, :-1],
iris_df.iloc[:, -1], test_size=0.2, random_state=0)
```

In this step, we've splitted the dataset into training and testing sets using train_test_split from scikit-learn.model_selection. The test_size argument specifies the fraction of the dataset to use for testing and the random_state argument sets a seed for reproducibility.

Step 4: Train the Model

```
# Train the Random Forest Classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=0)
classifier.fit(X_train, y_train)
```

In this step, we've created a RandomForestClassifier object and train it on the training data using the fit method. The n_estimators argument specifies the number of trees in the forest and the random_state argument sets a seed for reproducibility.

Step 5: Evaluate the Model

```
# Predict the class labels for the testing set
y_pred = classifier.predict(X_test)
# Evaluate the model performance using the confusion matrix and the
classification report
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)
print("Confusion Matrix:\n", cm)
print("\nClassification Report:\n", cr)
```

In this step, we've used the trained model and evaluate the performance.

Practice tasks

Load the iris dataset from scikit-learn and use (1) decision trees and (2) random forests to classify the iris species (setosa, versicolor, virginica) based on

a. sepal length and sepal width
b. petal length and petal width
c. sepal length, sepal width and petal length

You should split the data into training (80%) and testing (20%) sets, fit the model to the training data, make predictions on the test data, and evaluate the performance of the model using accuracy, precision, recall and F1-score metrics.