Experiment 1: Introduction to Popular Machine Learning Toolkits

Aim:

To understand and demonstrate the usage of popular machine learning toolkits: NumPy, Scikit-learn, Pandas, TensorFlow, and Keras.

Apparatus Required:

- Python (Anaconda/Miniconda/Standard Python Installation)
- Jupyter Notebook or any Python IDE
- Installed Libraries: numpy, pandas, scikit-learn, tensorflow, keras

Theory:

Machine learning relies heavily on various Python libraries that offer functions and utilities to ease the development process. Here are the commonly used libraries:

- **NumPy**: Provides support for large multi-dimensional arrays and matrices, along with mathematical functions to operate on them.
- **Pandas**: Offers data structures and operations for manipulating numerical tables and time series.
- **Scikit-learn**: Built on NumPy and SciPy, it provides simple and efficient tools for data mining and data analysis.
- **TensorFlow**: An end-to-end open-source platform for machine learning by Google.
- **Keras**: A high-level neural networks API, written in Python and capable of running on top of TensorFlow.

Code:

```
import numpy as np
print("NumPy version:", np.__version__)
print("\n")
print("\n")
x = np.array([[1,2,3],[4,5,6]])
print("x = {}\n".format(x))

import pandas as pd
print("Pandas version:", pd.__version__)
print("\n")
print("\n")
data = {'Name': ["John", "Anna", "Peter", "Linda"],
```

```
        'Location': ["New York", "India", "Paris", "Londan"],
         'Age': [24, 18, 40, 33]
        }
data_pandas = pd.DataFrame(data)
display(data_pandas)
display(data_pandas[data_pandas.Age > 30])

import sklearn
print("scikit-learn version:", sklearn.__version__)
print("\n")
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Load sample dataset
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3,
random_state=42)
# Train a simple model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
# Predict and evaluate
predictions = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, predictions) * 100, "%")

import tensorflow as tf
print("TensorFlow version:", tf.__version__)
!pip show tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

```
# Load and prepare data
iris = load_iris()
X = iris.data
y = to_categorical(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Build model
model = Sequential([
    Dense(10, input_shape=(4,), activation='relu'),
    Dense(3, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, verbose=0)
# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Test accuracy:", accuracy)

import keras
print("Standalone Keras version:", keras.__version__)
!pip show keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
iris = load_iris()
X = iris.data
y = to_categorical(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = Sequential([
    Dense(10, input_shape=(4,), activation='relu'),
    Dense(3, activation='softmax')
```

```
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, verbose=0)
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Test accuracy:", accuracy)
```

Result:

The versions of the machine learning toolkits were successfully displayed, confirming that the libraries are installed and can be imported without any issues. This validates the environment setup for future experiments involving machine learning workflows.