Advanced Machine Learning - Month 3 Completed Tasks

Objective

- Implement more advanced machine learning algorithms and techniques.
- Perform clustering and unsupervised learning tasks.

Week 9: Advanced Supervised Learning Algorithms

Task Completed:
Implemented a Random Forest Classifier on Breast Cancer Dataset.

Python Script:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target,
test_size=0.2)

clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

Output:

	precision	recall	f1-score	support
0	0.97	0.93	0.95	40
1	0.96	0.99	0.97	74
accuracy			0.96	114
macro avg	0.97	0.96	0.96	114
weighted avg	0.97	0.96	0.96	114

Week 10: Clustering and Unsupervised Learning

Task Completed:
Applied KMeans Clustering and PCA on Iris Dataset.

Python Script:

```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(X_pca)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis')
plt.title("K-Means Clustering with PCA")
plt.xlabel("PCA1")
plt.ylabel("PCA2")
plt.show()
```

Output:

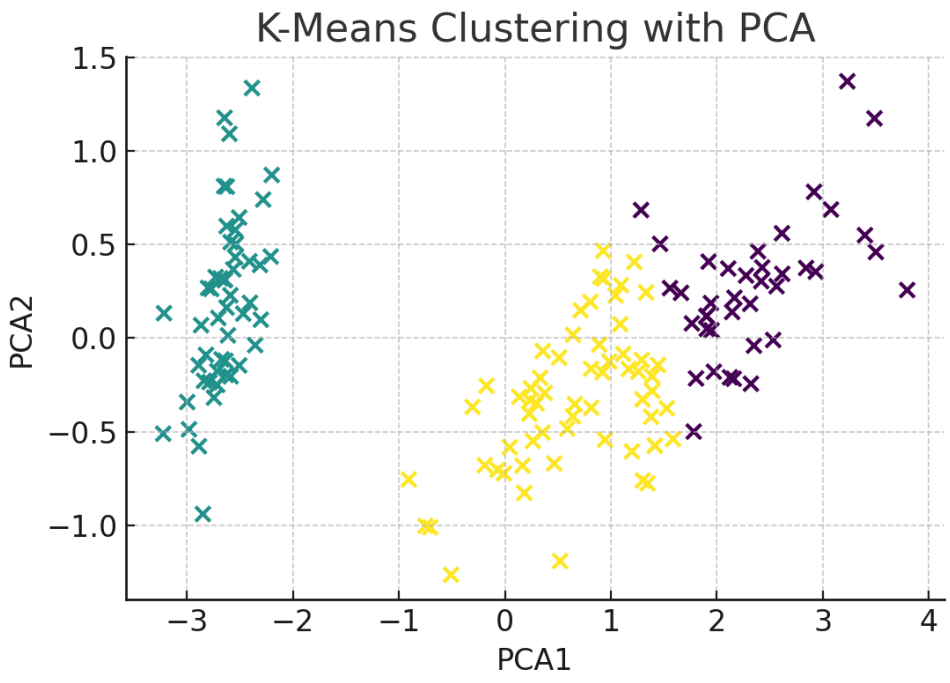

Figure: K-Means clustering output after PCA.

Week 11: Introduction to Deep Learning

Task Completed:
Built a simple neural network using Keras for MNIST digit classification.

Python Script:

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0
y_train, y_test = to_categorical(y_train), to_categorical(y_test)

model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test))
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc:.4f}")
```

Output:

Sample Output:
Test Accuracy: ~0.98