Lab Manual

B.Tech V

Deep Learning

1. Implementation of Python Basic Libraries (Statistics, Math, NumPy, SciPy)

Objective: Understand and use core scientific libraries in Python for mathematical and statistical computation.

```
python
CopyEdit
import math
import statistics
import numpy as np
from scipy import stats

# Math operations
print("Square root of 16:", math.sqrt(16))
print("Factorial of 5:", math.factorial(5))

# Statistics operations
data = [10, 20, 30, 40, 50]
print("Mean:", statistics.mean(data))
print("Median:", statistics.median(data))
print("Mode:", statistics.mode(data))

# NumPy operations
array = np.array(data)
print("Standard Deviation:", np.std(array))

# SciPy operations
print("Z-score:", stats.zscore(array))
```

2. WAP to Implement Union, Intersection, Complement

Objective: Perform set theory operations in Python.

```
python
CopyEdit
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}

print("Union:", A | B)
print("Intersection:", A & B)
print("Complement of A (w.r.t universal set U):", set(range(1, 7)) - A)
```

3. Using Pandas & Matplotlib for ML Applications

Objective: Load and visualize data using `pandas` and `matplotlib`.

```
python
CopyEdit
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-
data/master/iris.csv")
print(df.head())

# Plot
df['species'].value_counts().plot(kind='bar')
plt.title('Iris Species Count')
plt.xlabel('Species')
plt.ylabel('Count')
plt.show()
```

---

4. WAP to Implement an Artificial Neural Network

Objective: Build a simple ANN using `keras`.

```
python
CopyEdit
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

model = Sequential()
model.add(Dense(10, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X, y, epochs=100, verbose=0)

print("Prediction:", model.predict(X))
```

5. Creation and Loading of Different Datasets

Objective: Load built-in and custom datasets.

```
python
CopyEdit
from sklearn.datasets import load_iris, make_classification

# Built-in dataset
iris = load_iris()
print(iris.feature_names)

# Synthetic dataset
X, y = make_classification(n_samples=100, n_features=4, random_state=42)
print("Generated Data Shape:", X.shape)
```

6. WAP for Different Activation Functions

Objective: Implement and visualize activation functions.

```
python
CopyEdit
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-10, 10, 100)

relu = lambda x: np.maximum(0, x)
sigmoid = lambda x: 1 / (1 + np.exp(-x))
tanh = np.tanh

plt.plot(x, relu(x), label="ReLU")
plt.plot(x, sigmoid(x), label="Sigmoid")
plt.plot(x, tanh(x), label="Tanh")
plt.legend()
plt.title("Activation Functions")
plt.show()
```

7. Dimensionality Reduction using PCA (Iris Dataset)

Objective: Apply PCA on real-world data.

```
python
CopyEdit
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data
y = iris.target

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
plt.title("PCA on Iris")
plt.show()
```

---

8. WAP for Learning Rules (e.g., Hebbian Rule)

Objective: Implement Hebbian learning.

```
python
CopyEdit
import numpy as np

X = np.array([[1, 0], [0, 1]])
y = np.array([[1], [1]])

weights = np.zeros((2, 1))
for i in range(len(X)):
    weights += X[i].reshape(2, 1) * y[i]

print("Weights after Hebbian Learning:\n", weights)
```

---

9. WAP for Perceptron Network

Objective: Implement a single-layer perceptron.

```
python
CopyEdit
from sklearn.linear_model import Perceptron
import numpy as np

X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([0, 0, 0, 1])

clf = Perceptron()
clf.fit(X, y)

print("Predictions:", clf.predict(X))
```

10. Pattern Matching using Rules

Objective: Simple rule-based pattern matching.

```
python
CopyEdit
import re

patterns = [r'\bhello\b', r'\bworld\b']
text = "hello there! welcome to the world of AI."

for p in patterns:
    if re.search(p, text):
        print(f"Pattern '{p}' matched in text.")
```

---

12. Deep Learning in Healthcare

Objective: Use deep learning to classify diseases (e.g., diabetes dataset).

```
python
CopyEdit
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

data = load_diabetes()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target)

model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=100, verbose=0)

print("Model Evaluation:", model.evaluate(X_test, y_test))
```

13. Deep Learning in Business Analysis

Objective: Predict customer churn using ANN.

```
python
CopyEdit
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense

# Dummy DataFrame
data = pd.DataFrame({
    'age': [25,45,35,50,23],
    'salary': [50000, 64000, 58000, 70000, 52000],
    'churn': [0,1,0,1,0]
})

X = data[['age', 'salary']]
y = data['churn']

X_train, X_test, y_train, y_test = train_test_split(X, y)

model = Sequential()
model.add(Dense(10, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=100, verbose=0)

print("Churn Predictions:", model.predict(X_test))
```

15. Implement SVM with Different Kernels

Objective: Apply SVM with linear, RBF, and polynomial kernels.

```
python
CopyEdit
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

kernels = ['linear', 'rbf', 'poly']
for k in kernels:
    model = SVC(kernel=k)
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    print(f"Accuracy with {k} kernel: {accuracy_score(y_test, preds):.2f}")
```