Machine Learning Lab Exam - Answer Template

Use this template for each question in your exam

Question Number: ___

Aim:

Write the objective/goal of the experiment clearly and concisely.

Example:

- To implement a Decision Tree Classifier on the Iris dataset and visualize the tree structure.
- To predict housing prices using Linear Regression on the Boston Housing dataset.
- To implement FIND-S algorithm to find the most specific hypothesis from the given training data.

Software/Tool:

List the software, tools, and environment used.

Example:

- Google Colab
- Python 3.x
- Libraries: scikit-learn, pandas, numpy, matplotlib, seaborn

---

Algorithm/Procedure:

Write the step-by-step procedure or algorithm used to solve the problem.

Format:

1. Step 1: Description
2. Step 2: Description
3. Step 3: Description ... and so on

Example for Classification Problem:

1. Import required libraries
2. Load the dataset
3. Check for missing values and handle them
4. Split the data into training and testing sets (80-20)
5. Create and train the classifier model
6. Make predictions on test data
7. Calculate accuracy and display confusion matrix
8. Display results

Example for FIND-S Algorithm:

1. Initialize hypothesis with the most specific hypothesis (all nulls/φ)
2. For each positive training example:

    - If attribute value matches hypothesis, keep it
    - If attribute value differs, generalize to '?'
    - Skip negative examples
3. Display the final hypothesis
4. Return the maximally specific hypothesis

Program/Code:

python

```python
# Write your complete, well-commented code here

# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt

# Your code implementation
# Add comments for each major step

# Example structure:
# 1. Load dataset
# 2. Preprocessing
# 3. Model training
# 4. Evaluation
# 5. Display results
```

Important:

- Add comments explaining each section
- Use meaningful variable names
- Include all necessary imports
- Use random_state=42 where applicable

Output:

Display all the outputs generated by your program.

Include:

- Print statements showing data shapes, missing values, etc.
- Model accuracy scores
- Confusion matrix
- Classification reports

- Predictions (first 5-10 samples if applicable)
- Visualizations (plots, graphs, tree structures)
- Any other relevant output

Example Output Format:

Dataset Shape: (150, 4)
Training Set: (120, 4)
Testing Set: (30, 4)

Missing Values: 0

Model Accuracy: 96.67%

Confusion Matrix:
[[10 0 0]
[0 9 1]
[0 0 10]]

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	0.90	0.95	10
2	0.91	1.00	0.95	10

[Include any plots/visualizations as images]

Result:

Write a brief conclusion summarizing what was achieved.

Format:

- State what was successfully implemented
- Mention key findings (accuracy, important features, etc.)
- Brief interpretation of results

Example:

- Successfully implemented a Decision Tree Classifier on the Iris dataset with 96.67% accuracy.
- The model correctly classified all Setosa species and showed minor confusion between Versicolor and Virginica species.
- The decision tree visualization shows that petal width and petal length are the most important features for classification.

OR

- Successfully implemented the FIND-S algorithm on the EnjoySport dataset.

- The final hypothesis obtained is: <Sunny, Warm, ?, Strong, ?, ?>
- This hypothesis correctly generalizes all positive training examples.

Additional Notes:

1. **Be Concise:** Keep your aim and result brief and to the point
2. **Be Complete:** Include all code and output
3. **Be Clear:** Use proper formatting and comments
4. **Be Accurate:** Ensure output matches your code
5. **Be Neat:** Organize your answer using this template

Tips for Writing Good Answers:

✅ **Aim:** One or two clear sentences ✅ **Algorithm:** Step-by-step, numbered format ✅ **Code:** Well-commented, properly indented ✅ **Output:** All relevant results displayed ✅ **Result:** Brief summary with key metrics

---

End of Template

Sample Complete Answer Using This Template

Question Number: 1

Aim:

To build a k-NN classifier on the Iris dataset and find the optimal value of K by testing with K=3, 5, and 7.

Software/Tool:

- Google Colab
- Python 3.10
- Libraries: scikit-learn, pandas, numpy, matplotlib

Algorithm/Procedure:

1. Import required libraries (sklearn, pandas, numpy, matplotlib)
2. Load the Iris dataset using sklearn
3. Split the data into training (80%) and testing (20%) sets with random_state=42
4. Create three k-NN classifiers with K=3, K=5, and K=7
5. Train each model on the training data
6. Make predictions on test data for each model
7. Calculate accuracy for each K value
8. Plot accuracy vs K value graph
9. Identify the optimal K value
10. Display results and conclusion

Program/Code:

python

```python
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Dataset Shape:", X.shape)
print("Training Set Shape:", X_train.shape)
print("Testing Set Shape:", X_test.shape)
print()

# Test different K values
k_values = [3, 5, 7]
accuracies = []

for k in k_values:
    # Create k-NN classifier
    knn = KNeighborsClassifier(n_neighbors=k)

    # Train the model
    knn.fit(X_train, y_train)

    # Make predictions
    y_pred = knn.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)

    print(f"K = {k}: Accuracy = {accuracy*100:.2f}%")

# Plot accuracy vs K value
plt.figure(figsize=(8, 5))
```

```
plt.plot(k_values, accuracies, marker='o', linewidth=2, markersize=8)
plt.xlabel('K Value', fontsize=12)
plt.ylabel('Accuracy', fontsize=12)
plt.title('Accuracy vs K Value for k-NN Classifier', fontsize=14)
plt.grid(True)
plt.xticks(k_values)
plt.show()

# Find optimal K
optimal_k = k_values[accuracies.index(max(accuracies))]
print(f"\nOptimal K value: {optimal_k} with accuracy: {max(accuracies)*100:.2f}%")
```

Output:

```
Dataset Shape: (150, 4)
Training Set Shape: (120, 4)
Testing Set Shape: (30, 4)

K = 3: Accuracy = 100.00%
K = 5: Accuracy = 100.00%
K = 7: Accuracy = 96.67%

Optimal K value: 3 with accuracy: 100.00%
```

Graph: [A line graph showing accuracy on Y-axis and K values (3, 5, 7) on X-axis, with accuracy dropping slightly at K=7]

Result:

Successfully implemented k-NN classifier on the Iris dataset with three different K values. The model achieved 100% accuracy with both K=3 and K=5, and 96.67% accuracy with K=7. The optimal K value is 3, as it achieves the highest accuracy with the smallest neighborhood size, making it computationally efficient.

End of Sample Answer