Student Name:____________________________Roll No:__________________Section:_____________

CS334 - Machine Learning

Lab 13

Instructor: Ms. Maham Ashraf
E-mail: mashraf@uit.edu

Semester: Spring 2024

Objective

This labs objective is Understanding hyper parameter tuning and evaluating models performance

Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

How to Submit

- Submit lab work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Lab work file name should be saved with your roll number (e.g. 19a-001-SE_LW01.py)
- Submit home work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Home work file name should be saved with your roll number (e.g. 19a-001-SE_HW01.py)

Hyperparameter Tuning in Machine Learning:

Hyperparameter tuning is a crucial step in the process of building machine learning models, as it involves finding the optimal set of hyperparameters that govern the behavior of the model. Hyperparameters are parameters that are not learned from the training data but are set prior to the training process. They significantly impact the model's performance and generalization capabilities. Effective hyperparameter tuning can lead to improved model accuracy, robustness, and better handling of unseen data.

Importance of Hyperparameter Tuning:

Model Performance Improvement:
The choice of hyperparameters directly affects the model's performance metrics, such as accuracy, precision, recall, and F1-score. Tuning these parameters can lead to significant improvements in these metrics.
Generalization and Overfitting:
Hyperparameters play a crucial role in controlling the complexity of a model. Proper tuning helps strike a balance between underfitting and overfitting, promoting better generalization to new, unseen data.
Model Robustness:
Robust models should perform well across different datasets and scenarios. Hyperparameter tuning helps create models that are more adaptable to various data distributions, making them robust and reliable.
Resource Utilization:
Efficient hyperparameter tuning can help optimize computational resources. It allows practitioners to fine-tune models without exhaustive search, saving time and computational power.

Common Hyperparameters in Machine Learning:

Learning Rate (for gradient-based optimization algorithms):
A crucial hyperparameter in optimization algorithms like stochastic gradient descent. It determines the step size during optimization.
Number of Trees (for ensemble methods like Random Forest):
The number of trees in an ensemble affects model complexity and performance. Too few trees may lead to underfitting, while too many trees may lead to overfitting.
Depth and Split Criteria (for decision tree-based models):
The depth of decision trees and criteria for splitting nodes impact model complexity and decision-making. Tuning these hyperparameters can prevent trees from being too shallow or too deep.
Regularization Parameters (e.g., C in Support Vector Machines):
Control the trade-off between fitting the training data and preventing overfitting. Larger values may lead to more complex models.
Neural Network Architecture Hyperparameters (e.g., number of layers, number of units in each layer):
The architecture of neural networks significantly impacts their ability to learn complex patterns. Proper tuning can lead to better representation learning.

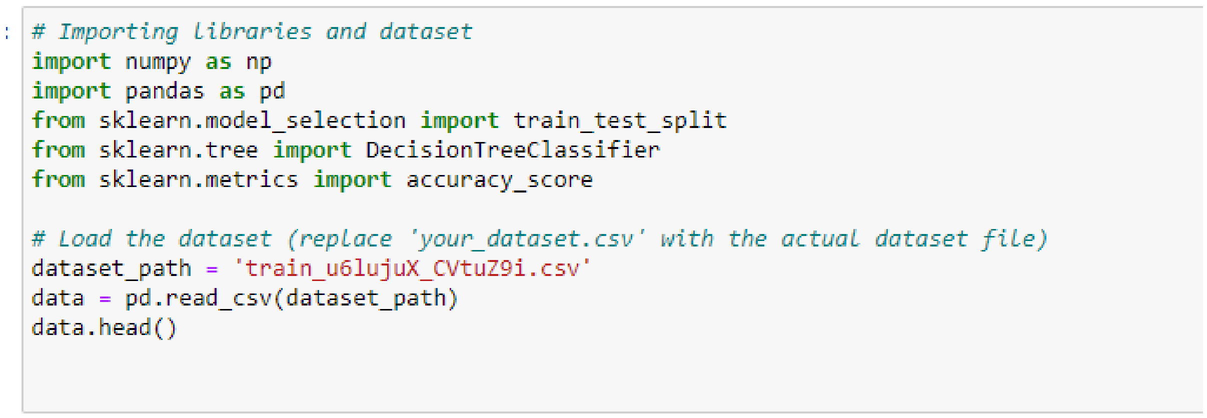
Hyperparameter Tuning Techniques:

1. **Grid Search:**
   Grid search is a systematic approach where a predefined grid of hyperparameter values is explored. It evaluates the model's performance for each combination using cross-validation or a validation set, helping to identify the optimal set of hyperparameters.
2. **Randomized Search:**
   Randomized search samples hyperparameter values randomly from predefined distributions. It efficiently explores the hyperparameter space without exhaustively trying all combinations, making it suitable for large search spaces.
3. **Bayesian Optimization:**
   Bayesian optimization leverages probabilistic models to model the performance landscape of a model with respect to hyperparameters. It makes informed decisions on which hyperparameter values to try next, efficiently converging to the optimal set.
4. **Manual Tuning:**
   Domain expertise and intuition play a role in hyperparameter tuning. Manual tuning involves iteratively adjusting hyperparameters based on knowledge of the problem and the behavior of the model.

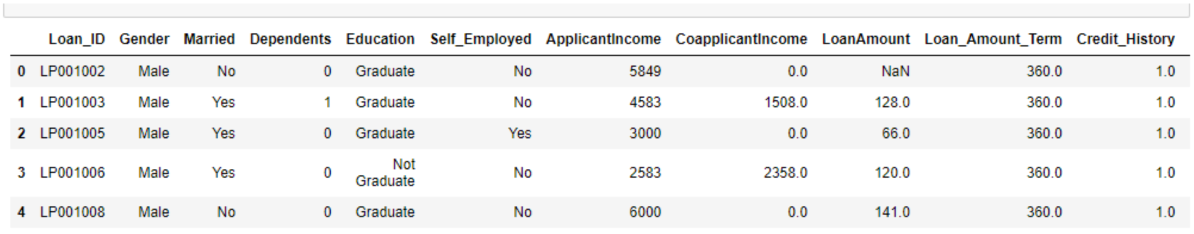
Workflow for Hyperparameter Tuning:

Data Preprocessing:
Handle missing values, encode categorical variables, and scale features appropriately.
Model Selection:
Choose a machine learning algorithm suitable for the problem at hand.
Split Data:
Divide the dataset into training, validation, and testing sets.
Define Hyperparameter Space:
Specify the hyperparameters to be tuned and their possible values or distributions.
Choose Tuning Technique:
Select a hyperparameter tuning technique based on the available computational resources and search efficiency.
Perform Tuning:
Apply the chosen technique to find the optimal hyperparameters by iteratively training and evaluating the model.
Evaluate on Test Set:
Assess the final model's performance on a separate test set to estimate its generalization capabilities.

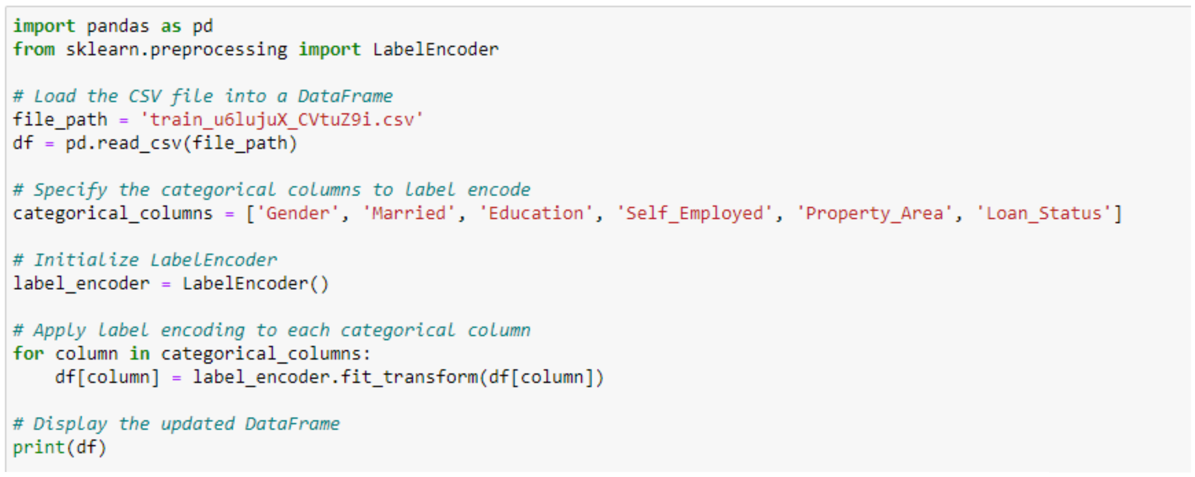
Implementation:



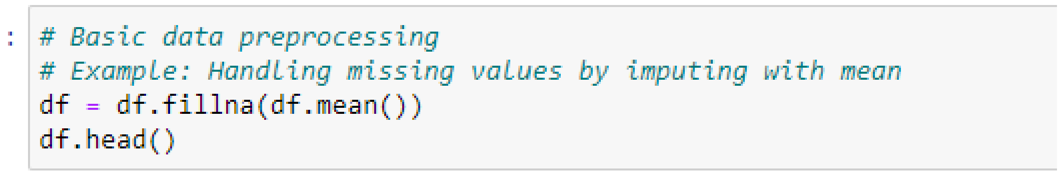
```
# Importing libraries and dataset
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the dataset (replace 'your_dataset.csv' with the actual dataset file)
dataset_path = 'train_u6lujuX_CVtuZ9i.csv'
data = pd.read_csv(dataset_path)
data.head()
```



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0



```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Load the CSV file into a DataFrame
file_path = 'train_u6lujuX_CVtuZ9i.csv'
df = pd.read_csv(file_path)

# Specify the categorical columns to label encode
categorical_columns = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status']

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to each categorical column
for column in categorical_columns:
    df[column] = label_encoder.fit_transform(df[column])

# Display the updated DataFrame
print(df)
```



```
# Basic data preprocessing
# Example: Handling missing values by imputing with mean
df = df.fillna(df.mean())
df.head()
```

```
# Splitting data into features (X) and target variable (y)
X = df.drop('Loan_Status', axis=1)  # Adjust 'target_variable_column' to your dataset
y = df['Loan_Status']

# Splitting 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)

# Implementing a simple machine learning model (e.g., Decision Tree)
# Instantiate the Decision Tree classifier
model = DecisionTreeClassifier(random_state=42)

# Fit the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of the Decision Tree model:", accuracy)
```

```
Accuracy of the Decision Tree model: 0.6910569105691057
```

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV


param_grid = {'criterion': ['gini', 'entropy'],
              'splitter': ['best', 'random'],
              'max_depth': [None, 10, 20, 30, 40, 50],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4]}

param_dist = {'criterion': ['gini', 'entropy'],
              'splitter': ['best', 'random'],
              'max_depth': [None, 10, 20, 30, 40, 50],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4]}
```

```
# Grid Search
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Random Search
random_search = RandomizedSearchCV(model, param_distributions=param_dist, n_iter=10, cv=5)
random_search.fit(X_train, y_train)


# Evaluate models on the testing set
y_pred_grid = grid_search.predict(X_test)
y_pred_random = random_search.predict(X_test)

# Calculate accuracy scores
accuracy_grid = accuracy_score(y_test, y_pred_grid)
accuracy_random = accuracy_score(y_test, y_pred_random)

# Print accuracy scores
print("Accuracy with Grid Search:", accuracy_grid)
print("Accuracy with Random Search:", accuracy_random)
```

```
Accuracy with Grid Search: 0.7235772357723578
Accuracy with Random Search: 0.7642276422764228
```

Lab Tasks:

Task 1: Data Loading and Hyperparameter Tuning

1. **Data Loading:**
    - **Question 1:** Load a dataset of your choice (e.g., CSV, Excel) into a pandas DataFrame. What steps would you take to ensure the proper loading and exploration of the data?
    - **Question 2:** Explore and briefly describe the loaded dataset. What are the features, and what type of machine learning task could be performed with this data?
2. **Data Preprocessing:**
    - **Question 3:** Preprocess the data as needed for a classification task. Handle any missing values, encode categorical variables, and split the dataset into training and testing sets.
    - **Question 4:** Why is it important to preprocess the data before training a machine learning model?
3. **Hyperparameter Tuning with Grid Search:**
    - **Question 5:** Choose a classification model (e.g., Decision Tree, Random Forest). Define a hyperparameter grid for Grid Search. Which hyperparameters will you include, and what ranges/values will you explore?
    - **Question 6:** Implement Grid Search on the training set and identify the best hyperparameters. Train the final model and evaluate its performance on the testing set.
    - **Question 7:** Reflect on the impact of hyperparameter tuning. How did it influence the model's performance?

Task 2: Hyperparameter Tuning with other technique

1. **Model Selection:**
    - **Question 1:** Choose a different classification model for this task. Why did you choose this model, and what makes it suitable for your data?
    - **Question 2:** Briefly explain the hyperparameters of the chosen model that might significantly influence its performance.
2. **Hyperparameter Tuning with Randomized Search:**
    - **Question 3:** Define hyperparameter distributions for this technique. How does this differ from Grid Search?
    - **Question 4:** Implement other technique on the training set, identifying the best hyperparameters. Train the final model and evaluate its performance on the testing set.
    - **Question 5:** Compare the results obtained with the model's default hyperparameters. Discuss any observed differences and the efficiency
3. **Reflection:**
    - **Question 6:** Reflect on the experience of performing hyperparameter tuning with both Grid Search and other technique. Which method did you find more efficient, and why?