Assignment Code: DA-AG-015

Boosting Techniques | **Assignment**

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

Question 1: What is Boosting in Machine Learning? Explain how it improves weak learners.

Answer:

Boosting is an ensemble learning technique in machine learning where **multiple weak learners are trained sequentially**, with each new model focusing on **correcting the errors of the previous ones**. A weak learner is a model that performs slightly better than random guessing.

How Boosting Works to Improve Weak Learners

1. **Train the first weak learner** on the original data.
2. **Identify errors** made by this model (which data points it predicted incorrectly).
3. **Give more weight** to the misclassified examples so that the next model focuses on them.
4. **Train the next weak learner** on the weighted data.
5. Repeat this process sequentially, adding learners one by one.
6. **Combine all learners** (often using a weighted sum or voting) to make the final prediction.

Key Idea

Boosting converts a set of **weak learners** into a **strong learner**. Each model corrects the mistakes of the previous ones, so the overall ensemble becomes increasingly accurate.

Popular Boosting Algorithms

- **AdaBoost**
- **Gradient Boosting**
- **XGBoost**
- **LightGBM**

Question 2: What is the difference between AdaBoost and Gradient Boosting in terms of how models are trained?

Answer:

1. AdaBoost (Adaptive Boosting)

- Models are trained **sequentially**, with each new model focusing on the **misclassified samples** from the previous model.
- It adjusts the **weights of data points**: misclassified points get higher weights so that the next learner pays more attention to them.
- The final prediction is a **weighted vote** of all weak learners.

---

2. Gradient Boosting

- Models are also trained **sequentially**, but instead of reweighting data points, each new model is trained to **predict the residual errors** (differences between actual and predicted values) of the previous model.
- It uses **gradient descent** to minimize a specified **loss function** (e.g., mean squared error for regression).
- The final prediction is the **sum of predictions** from all models.

Question 3: How does regularization help in XGBoost?

Answer:

In XGBoost, **regularization** plays a key role in improving model generalization and preventing overfitting. It adds a penalty for model complexity to keep the trees simple and robust.

Intuitively, regularization helps XGBoost avoid learning noise from the training data by:

- **Penalizing large leaf weights**, so the model doesn't rely too heavily on any single feature or rule.

- **Discouraging too many tree splits**, ensuring the model focuses on meaningful patterns instead of memorizing data.

Question 4: Why is CatBoost considered efficient for handling categorical data?

Answer:

CatBoost is considered efficient for handling categorical data because it **natively supports categorical features** without requiring extensive preprocessing like one-hot encoding

How CatBoost Handles Categorical Data

1. **Automatic Encoding:** CatBoost converts categorical features into numerical representations internally using **ordered target statistics** or **frequency-based encoding**, reducing the risk of data leakage.
2. **Preserves Feature Relationships:** Unlike one-hot encoding, CatBoost retains the **relationship between categories** and the target variable, which helps the model capture patterns more effectively.
3. **Reduces Overfitting:** Its encoding method combined with **ordered boosting** prevents overfitting, even on small datasets.
4. **Efficient Computation:** CatBoost optimizes computations for categorical features, making it faster and more memory-efficient than manually encoding large categorical variables.

.

Question 5: What are some real-world applications where boosting techniques are preferred over bagging methods?

Answer:

Boosting techniques are preferred in applications needing **high accuracy**, **handling of complex patterns**, and **imbalanced data**, such as:

- **Fraud detection & credit scoring** – captures rare events and subtle patterns.
- **Ad click-through prediction** – models complex user–ad interactions.
- **Customer churn prediction** – handles mixed data types well.
- **Medical diagnosis** – robust with noisy, limited data.
- **Search ranking & recommendations** – optimizes ranking metrics directly.

Datasets:

- Use `sklearn.datasets.load_breast_cancer()` for classification tasks.
- Use `sklearn.datasets.fetch_california_housing()` for regression tasks.

Question 6: Write a Python program to:

- Train an AdaBoost Classifier on the Breast Cancer dataset
- Print the model accuracy

(*Include your Python code and output in the code box below.*)
Answer:

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

# Split the dataset 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)

# Initialize a base estimator (e.g., a Decision Tree Classifier)
# AdaBoost often uses shallow trees as base estimators
base_estimator = DecisionTreeClassifier(max_depth=1, random_state=42)

# Train an AdaBoost Classifier
adaboost_classifier = AdaBoostClassifier(estimator=base_estimator,
n_estimators=50, random_state=42)
adaboost_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred_adaboost = adaboost_classifier.predict(X_test)

# Calculate and print the model accuracy



accuracy_adaboost = accuracy_score(y_test, y_pred_adaboost)
print(f"Accuracy of the AdaBoost Classifier: {accuracy_adaboost:.4f}")
```

```
# output
Accuracy of the AdaBoost Classifier: 0.9649
```

Question 7: Write a Python program to:

- Train a Gradient Boosting Regressor on the California Housing dataset
- Evaluate performance using R-squared score

(*Include your Python code and output in the code box below.*)

Answer:

```
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score

# Load the California Housing dataset
housing = fetch_california_housing()
X = pd.DataFrame(housing.data, columns=housing.feature_names)
y = housing.target

# Split the dataset 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)

# Train a Gradient Boosting Regressor
gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
max_depth=3, random_state=42)
gbr.fit(X_train, y_train)

# Make predictions on the test set
y_pred_gbr = gbr.predict(X_test)
```

```
# Evaluate performance using R-squared score
r2 = r2_score(y_test, y_pred_gbr)
print(f"R-squared score for Gradient Boosting Regressor: {r2:.4f}")


# output
R-squared score for Gradient Boosting Regressor: 0.7756
```

Question 8: Write a Python program to:

- Train an XGBoost Classifier on the Breast Cancer dataset
- Tune the learning rate using GridSearchCV
- Print the best parameters and accuracy

(*Include your Python code and output in the code box below.*)
Answer:

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

# Split the dataset 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)

# Define the XGBoost Classifier
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
random_state=42)

# Define the hyperparameter grid for learning_rate tuning
param_grid = {
    'learning_rate': [0.01, 0.05, 0.1, 0.2]
}

# Perform GridSearchCV
```

```
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5,
scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Print the best parameters
print("Best parameters found:", grid_search.best_params_)

# Get the best estimator and evaluate on the test set
best_xgb_classifier = grid_search.best_estimator_
y_pred = best_xgb_classifier.predict(X_test)
final_accuracy = accuracy_score(y_test, y_pred)
print(f"Final accuracy on the test set with best parameters:
{final_accuracy:.4f}")

#output
```

```
/usr/local/lib/python3.12/dist-packages/xgboost/training.py:199:
UserWarning: [09:21:47] WARNING: /workspace/src/learner.cc:790:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
Best parameters found: {'learning_rate': 0.2}
Final accuracy on the test set with best parameters: 0.9561
```

Question 9: Write a Python program to:

- Train a CatBoost Classifier
- Plot the confusion matrix using seaborn

(*Include your Python code and output in the code box below.*)
Answer:

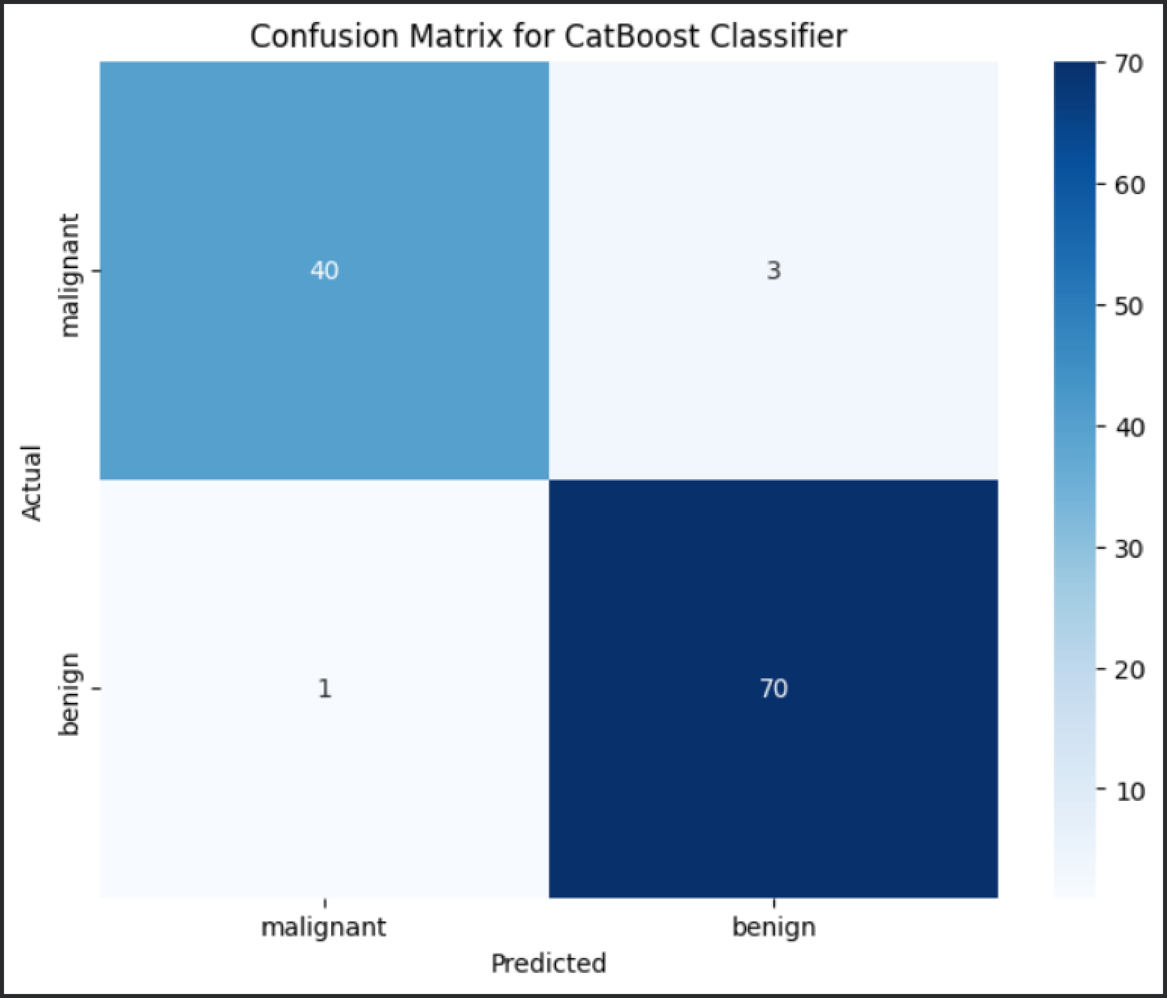
```
!pip install catboost

import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

# Split the dataset 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)

# Initialize and train a CatBoost Classifier
# verbose=0 suppresses training output for cleaner execution
cat_classifier = CatBoostClassifier(iterations=100,
                                    learning_rate=0.1,
                                    depth=6,
                                    loss_function='Logloss',
                                    eval_metric='Accuracy',
                                    random_seed=42,
                                    verbose=0)
cat_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred_catboost = cat_classifier.predict(X_test)

# Calculate accuracy (optional, but good for context)
```

```python
accuracy_catboost = accuracy_score(y_test, y_pred_catboost)
print(f"Accuracy of the CatBoost Classifier: {accuracy_catboost:.4f}")

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_catboost)

# Plot the confusion matrix using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=data.target_names, yticklabels=data.target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for CatBoost Classifier')
plt.show()


#output
```

```
Collecting catboost
  Downloading catboost-1.2.8-cp312-cp312-manylinux2014_x86_64.whl.metadata
(1.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.12/dist-
packages (from catboost) (0.21)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-
packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in
/usr/local/lib/python3.12/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in
/usr/local/lib/python3.12/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-
packages (from catboost) (1.16.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-
packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-
packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost)
(2.9.0.post0)
```

```
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost)
(2025.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.3.3)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.4.9)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-
packages (from matplotlib->catboost) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (3.2.5)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.12/dist-packages (from plotly->catboost) (8.5.0)
Downloading catboost-1.2.8-cp312-cp312-manylinux2014_x86_64.whl (99.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
99.2/99.2 MB 8.6 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.8
Accuracy of the CatBoost Classifier: 0.9649
```

```
Successfully installed catboost-1.2.8
Accuracy of the CatBoost Classifier: 0.9649
```



Question 10: You're working for a FinTech company trying to predict loan default using customer demographics and transaction behavior.

The dataset is imbalanced, contains missing values, and has both numeric and categorical features.

Describe your step-by-step data science pipeline using boosting techniques:

- Data preprocessing & handling missing/categorical values
- Choice between AdaBoost, XGBoost, or CatBoost
- Hyperparameter tuning strategy
- Evaluation metrics you'd choose and why
- How the business would benefit from your model

(*Include your Python code and output in the code box below.*)

Answer:

Here's a structured step-by-step pipeline for predicting **loan default** using boosting techniques in a real-world FinTech scenario:

---

1. Data Preprocessing

a. Handling missing values

- For **numeric features**: Impute with median or mean, or use model-based imputation if patterns exist.
- For **categorical features**: Impute with the most frequent category or a separate "Unknown" category.

b. Handling categorical features

- Since boosting algorithms handle categorical data differently:

- **CatBoost:** Can directly handle categorical features, no need for one-hot encoding.
- **XGBoost/AdaBoost:** Convert categorical features using **one-hot encoding** or **target encoding**.

c. Handling imbalanced data

- Since loan default is rare, consider:
- Using **class weights** in the loss function.
- **SMOTE** or oversampling techniques on the minority class. ○ Ensuring evaluation metrics are suitable for imbalanced datasets. **d. Feature scaling**
- Not necessary for tree-based boosting methods like XGBoost or CatBoost, but useful for certain numeric transformations if needed.

---

2. Choosing the Boosting Algorithm

- **CatBoost:** Best if dataset has many categorical features and missing values.
- **XGBoost:** Highly accurate, fast, and widely used; needs preprocessing for categorical features.
- **AdaBoost:** Simpler, works best with weak learners; less robust to noisy or imbalanced data.

Choice: CatBoost is preferred for this scenario due to native handling of categorical data, missing values, and robustness with imbalanced data.

---

3. Hyperparameter Tuning

- Use **Grid Search** or **Randomized Search** with **cross-validation**.
- Key parameters to tune for CatBoost/XGBoost:

o `learning_rate` (controls step size for each tree) o `depth` (max depth of trees)

o `iterations` / `n_estimators` (number of trees) o `l2_leaf_reg` (regularization to reduce overfitting) o `scale_pos_weight` (to handle class imbalance)

o

- Consider **early stopping** to prevent overfitting.

---

4. Evaluation Metrics

Since the dataset is imbalanced, accuracy alone is not sufficient. Use:

- **AUC-ROC**: Measures the model's ability to distinguish defaulters from nondefaulters.
- **Precision & Recall**: Recall is critical because missing a default is costly; precision ensures false positives are controlled.
- **F1-Score**: Balances precision and recall.
- Optionally, use **confusion matrix** or **Precision-Recall curve** for detailed insights.

---

5. Business Benefits

- **Risk Reduction:** Better identification of high-risk borrowers reduces loan default losses.
- **Targeted Decision-Making:** Insights from feature importance help design risk-based pricing or credit limits.
- **Operational Efficiency:** Automation of loan approval decisions using a robust predictive model.

- **Regulatory Compliance:** Transparent, explainable boosting models (especially CatBoost with SHAP values) help meet audit requirements.

In short:

Preprocess missing and categorical data → Choose CatBoost → Tune hyperparameters with cross-validation → Evaluate using recall, precision, AUC → Deploy model to reduce defaults and improve risk-based lending decisions.