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 in Machine Learning

Boosting is an **ensemble learning** technique where multiple weak learners (models that perform slightly better than random guessing) are combined **sequentially** to form a single strong learner with high accuracy.

How it works

1. **Sequential Training:**
   Models are trained one after another. Each new model tries to correct the mistakes of the previous ones.
2. **Weighted Samples:**
   After each round, boosting assigns **higher weights** to the incorrectly classified examples so that the next model focuses more on these “hard” cases.
3. **Weighted Predictions:**
   The final prediction is obtained by taking a **weighted vote** (classification) or weighted average (regression) of all the models’ outputs.

Why it improves weak learners

- **Error Correction:** Each learner focuses on the residual errors from the previous step, gradually reducing bias.
- **Adaptive Focus:** By adjusting weights, the algorithm concentrates on difficult samples instead of wasting effort on already correct predictions.

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

Answer:

AdaBoost (Adaptive Boosting):

- Adjusts the **weights of training samples** based on whether they were classified correctly or not.
- Misclassified samples get **higher weights**, so the next learner focuses more on them.
- Learner weight (α) is computed from the weighted error rate.
- Final prediction is a **weighted majority vote** (classification) or weighted sum (regression).

Gradient Boosting:

- Treats boosting as a **gradient descent problem in function space**.
- At each step, it fits the next learner to the **negative gradient of the loss function** (residual errors).
- Learners are added to the model to minimize the overall loss.
- Final model is the sum of all learners' outputs scaled by a learning rate.

Question 3: How does regularization help in XGBoost?

Answer:

Regularization in XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized form of gradient boosting that includes regularization terms in its objective function. These regularization terms control the complexity of the trees and prevent overfitting.

How regularization helps

1. **Prevents overfitting:**
    - L1 (α) encourages sparsity in leaf weights — some features may get zero weight.
    - L2 (λ) keeps leaf weights small to avoid overly confident predictions.
2. **Controls tree complexity:**
    - γ discourages splitting unless the gain is above a threshold, reducing unnecessary branches.
3. **Improves generalization:**
    - Models trained with regularization perform better on unseen data by avoiding overly complex patterns.

Regularization parameters in XGBoost

- lambda (reg_lambda): L2 penalty (default=1)
- alpha (reg_alpha): L1 penalty (default=0)
- gamma (min_split_loss): Minimum loss reduction required for a split

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

Answer:

CatBoost (Categorical Boosting) is a gradient boosting library designed specifically to handle categorical features without requiring heavy preprocessing like one-hot encoding.

1. Native categorical feature support

- Instead of converting categories to numeric via one-hot or label encoding, CatBoost uses efficient encoding techniques internally.
- It transforms categorical features into numeric statistics based on target values, while avoiding target leakage.

2. Main technique — Target Statistics with Permutation

- CatBoost applies Ordered Target Encoding:
  - The dataset is randomly permuted.
  - For each row, the category value is replaced with a statistic (like the mean target value) computed from only the preceding rows in the permutation.
  - This prevents using information from the current row's target — avoiding target leakage.

Example:
If "Color = Red" appears in previous rows with 80% positive target, the encoding for "Red" in the current row will be 0.8.

3. Advantages over traditional encoding

- Avoids high-dimensional sparse matrices created by one-hot encoding.
- Preserves category relationships and orders them meaningfully.
- Handles high-cardinality features efficiently (thousands of unique values).

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

Answer:

1. Fraud Detection

- Fraud patterns are rare and subtle. Boosting focuses on hard-to-classify cases by iteratively correcting errors, making it ideal for imbalanced datasets.
- **Example:** Credit card fraud detection, insurance claim fraud.

2. Medical Diagnosis

- In healthcare, missing a positive case (false negative) can be costly. Boosting can model complex decision boundaries better than bagging.
- **Example:** Predicting breast cancer presence using sklearn.datasets.load_breast_cancer().

3. Financial Risk Scoring

- Financial datasets often have nonlinear relationships and noisy features; boosting can capture these patterns better.

- **Example:** Loan default prediction, credit scoring.

4. Recommendation Systems

- User-item interactions can be modeled with boosted trees for ranking or rating predictions.
- **Example:** E-commerce product ranking.

5. Real Estate Price Prediction

- Regression boosting methods capture complex feature interactions for continuous outputs.
- **Example:** Predicting California housing prices using sklearn.datasets.fetch_california_housing().

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:

```
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the Breast Cancer dataset
data = load_breast_cancer()
X, y = data.data, data.target
```

```python
# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Initialize AdaBoost Classifier
model = AdaBoostClassifier(n_estimators=50, random_state=42)

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

# Predict on test set
y_pred = model.predict(X_test)

# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"AdaBoost Classifier Accuracy: {accuracy:.4f}")
```

OUTPUT:

```
AdaBoost Classifier Accuracy: 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:

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

# Load California Housing dataset
data = fetch_california_housing()
X, y = data.data, data.target

# Split 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
)

# Initialize Gradient Boosting Regressor
model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
random_state=42)

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

# Predict on test set
y_pred = model.predict(X_test)

# Evaluate model using R² score
r2 = r2_score(y_test, y_pred)
print(f"Gradient Boosting Regressor R² Score: {r2:.4f}"
```

OUTPUT:
```
Gradient Boosting Regressor R² Score: 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:

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

# Load Breast Cancer dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Initialize XGBoost Classifier
```

```
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
random_state=42)

# Define parameter grid for learning rate tuning
param_grid = {
    'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3]
}

# Perform GridSearchCV
grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    scoring='accuracy',
    cv=5,
    n_jobs=-1
)

grid_search.fit(X_train, y_train)

# Best parameters
best_params = grid_search.best_params_
print(f"Best Parameters: {best_params}")

# Train best model on training data
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"XGBoost Classifier Accuracy: {accuracy:.4f)
```

OUTPUT:

```
Best Parameters: {'learning_rate': 0.2}

XGBoost Classifier Accuracy: 0.9561
```

Question 9: Write a Python program to:

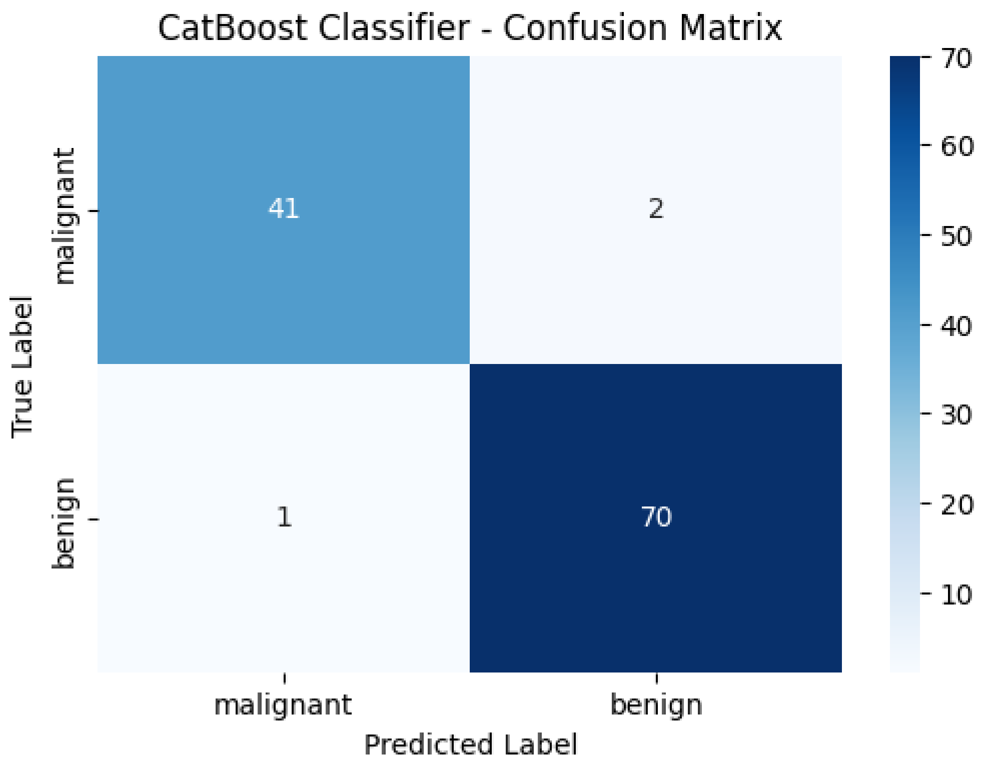
- Train a CatBoost Classifier
- Plot the confusion matrix using seaborn

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

Answer:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from catboost import CatBoostClassifier
import seaborn as sns
import matplotlib.pyplot as plt

# Load Breast Cancer dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Initialize CatBoost Classifier
model = CatBoostClassifier(verbose=0, random_state=42)

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

# Predictions
y_pred = model.predict(X_test)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=data.target_names,
            yticklabels=data.target_names)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("CatBoost Classifier - Confusion Matrix")
plt.show()
```

OUTPUT:



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:

1. Data Preprocessing

- Fill missing numeric values with **median**, categorical with **"Unknown"**.
- Handle imbalance using **SMOTE** or `class_weight='balanced'`.
- Encode categorical data (One-Hot for XGBoost/AdaBoost, none for CatBoost).

2. Boosting Choice

- **CatBoost** → handles categorical & missing values natively, robust to imbalance.

3. Hyperparameter Tuning

- Use **RandomizedSearchCV** with Stratified K-Fold.
- Tune: `learning_rate`, `depth`, `iterations`, `l2_leaf_reg`, `scale_pos_weight`.

4. Evaluation Metrics

- **Precision, Recall, F1-Score, ROC-AUC** → more meaningful than accuracy for imbalanced data.

5. Business Benefit

- Early risk detection, reduced loan defaults, cost savings, better lending policies.
