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 that combines multiple **weak learners** to create a **strong predictive model**. A weak learner is a model that performs only slightly better than random guessing.

In boosting, models are trained **sequentially**, where each new model focuses more on the **errors made by previous models**. Misclassified data points are given **higher importance (weights)** so that subsequent learners try harder to correct them.

By combining many weak learners and adjusting their influence, boosting **reduces bias and improves accuracy**, resulting in a strong and robust model.

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

Answer:

Feature	AdaBoost	Gradient Boosting
Training approach	Sequential	Sequential
Error handling	Increases weight of misclassified samples	Fits new model to residual errors
Loss function	Uses exponential loss	Can use any differentiable loss function
Flexibility	Less flexible	More flexible and powerful
Sensitivity to noise	Highly sensitive	Less sensitive (with tuning)

Question 3: How does regularization help in XGBoost?

Answer:

Regularization in XGBoost helps to **prevent overfitting** by penalizing overly complex models.

XGBoost uses:

- **L1 regularization (Lasso)** on leaf weights
- **L2 regularization (Ridge)** on leaf weights

It also controls complexity using:

- Maximum tree depth
- Minimum child weight
- Subsampling of rows and features

These regularization techniques help XGBoost produce **simpler, more generalized models**, improving performance on unseen 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 manual encoding.

Key reasons:

- Uses **ordered target encoding** to avoid data leakage
- Automatically converts categorical variables into numerical form
- Handles missing values internally
- Reduces overfitting caused by traditional encoding methods

Due to these features, CatBoost performs well on datasets with **many categorical variables** and requires **less preprocessing effort**.

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

Answer:
Boosting techniques are preferred when high accuracy and bias reduction are required .
Real-world applications include:

- **Fraud detection in banking and finance**
- **Credit risk and loan default prediction**
- **Spam email detection**
- **Medical diagnosis (cancer detection, disease prediction)**
- **Customer churn prediction**
- **Search ranking and recommendation systems**

Boosting is chosen over bagging when the dataset is complex and weak models need to be improved iteratively.

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:Real-world applications include:

- **Fraud detection in banking and finance**
- **Credit risk and loan default prediction**
- **Spam email detection**
- **Medical diagnosis (cancer detection, disease prediction)**
- **Customer churn prediction**
- **Search ranking and recommendation systems**

Boosting is chosen over bagging when the dataset is complex and weak models need to be improved iteratively.

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

# Load the dataset
data = load_breast_cancer()
X = data.data
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
)
```

```
# Train AdaBoost Classifier
ada = AdaBoostClassifier(n_estimators=100, random_state=42)
ada.fit(X_train, y_train)

# Make predictions
y_pred = ada.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print("Model Accuracy:", accuracy)

OUTPUT
Model Accuracy: 0.9561
```

Question 7: Write a Python program to:

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

```
# Import required libraries
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
data = fetch_california_housing()
X = data.data
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
)

# Train Gradient Boosting Regressor
gbr = GradientBoostingRegressor(random_state=42)
gbr.fit(X_train, y_train)

# Make predictions
y_pred = gbr.predict(X_test)
```

```
# Calculate R-squared score
r2 = r2_score(y_test, y_pred)

print("R-squared Score:", r2)

OUTPUT
R-squared Score: 0.7756446042829697
```

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 required libraries
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 the Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

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

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

# GridSearchCV
```

```
grid = GridSearchCV(
    estimator=xgb,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy'
)

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

# Get best model
best_model = grid.best_estimator_

# Make predictions
y_pred = best_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Print results
print("Best Parameters:", grid.best_params_)
print("Model Accuracy:", accuracy)
```

Best Parameters: {'learning_rate': 0.1}
Model Accuracy: 0.9649

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:**

```
# Import required libraries
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 the Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Train CatBoost Classifier
model = CatBoostClassifier(
    iterations=100,
    learning_rate=0.1,
    depth=6,
    verbose=False,
    random_state=42
)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Create confusion matrix
cm = confusion_matrix(y_test, y_pred)

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

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:

```
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Train CatBoost Classifier
model = CatBoostClassifier(
    iterations=300,
    learning_rate=0.05,
    depth=6,
    eval_metric='AUC',
    verbose=False
)

model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]
```

```
# Evaluation
print(classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_prob))
```