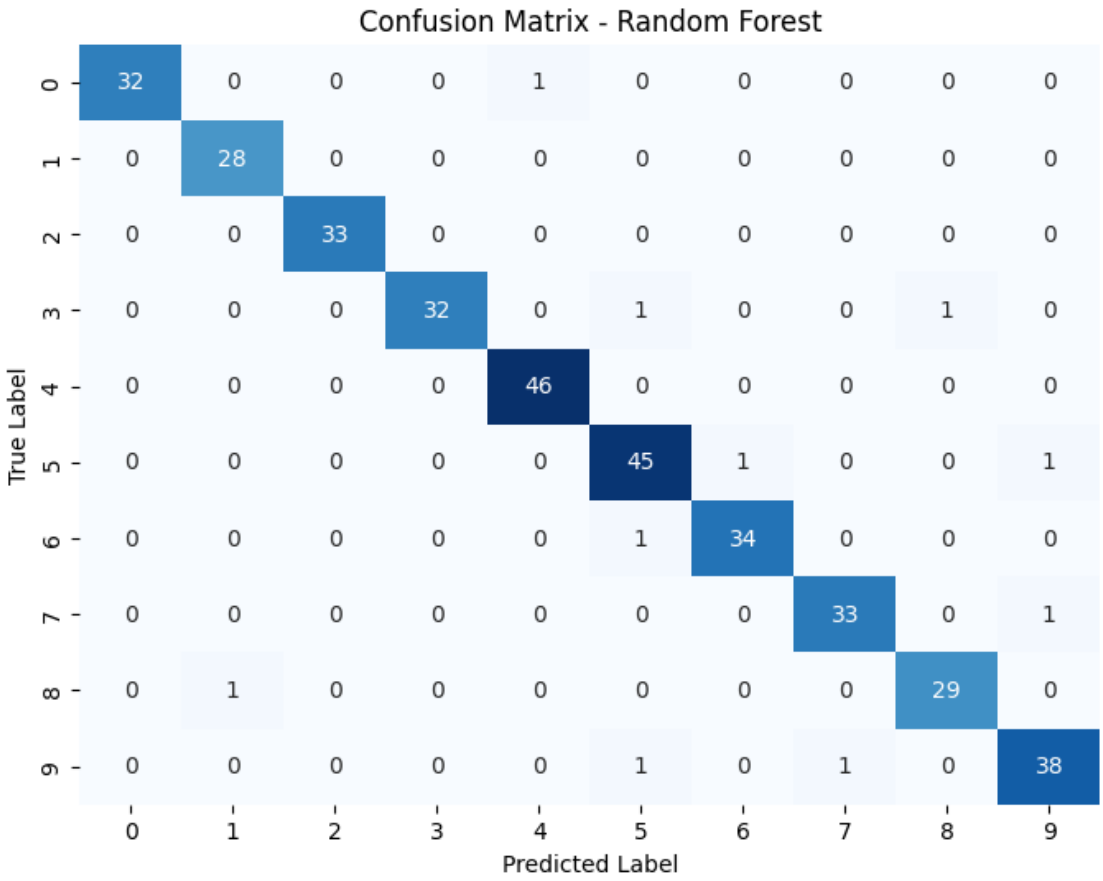
day21

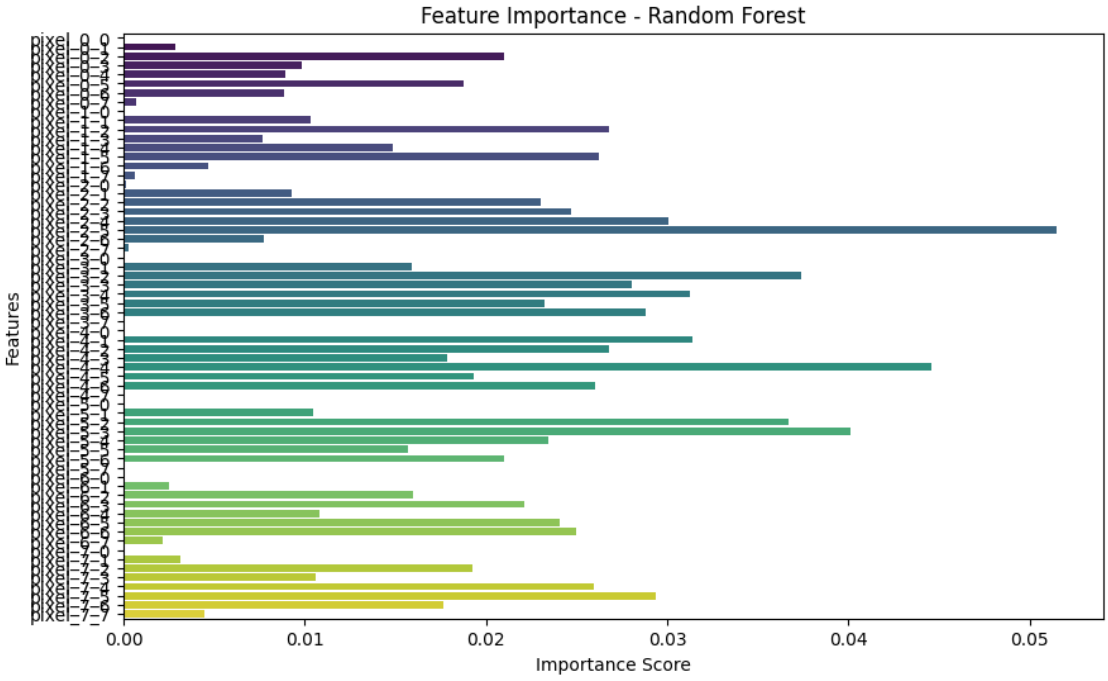
February 28, 2024

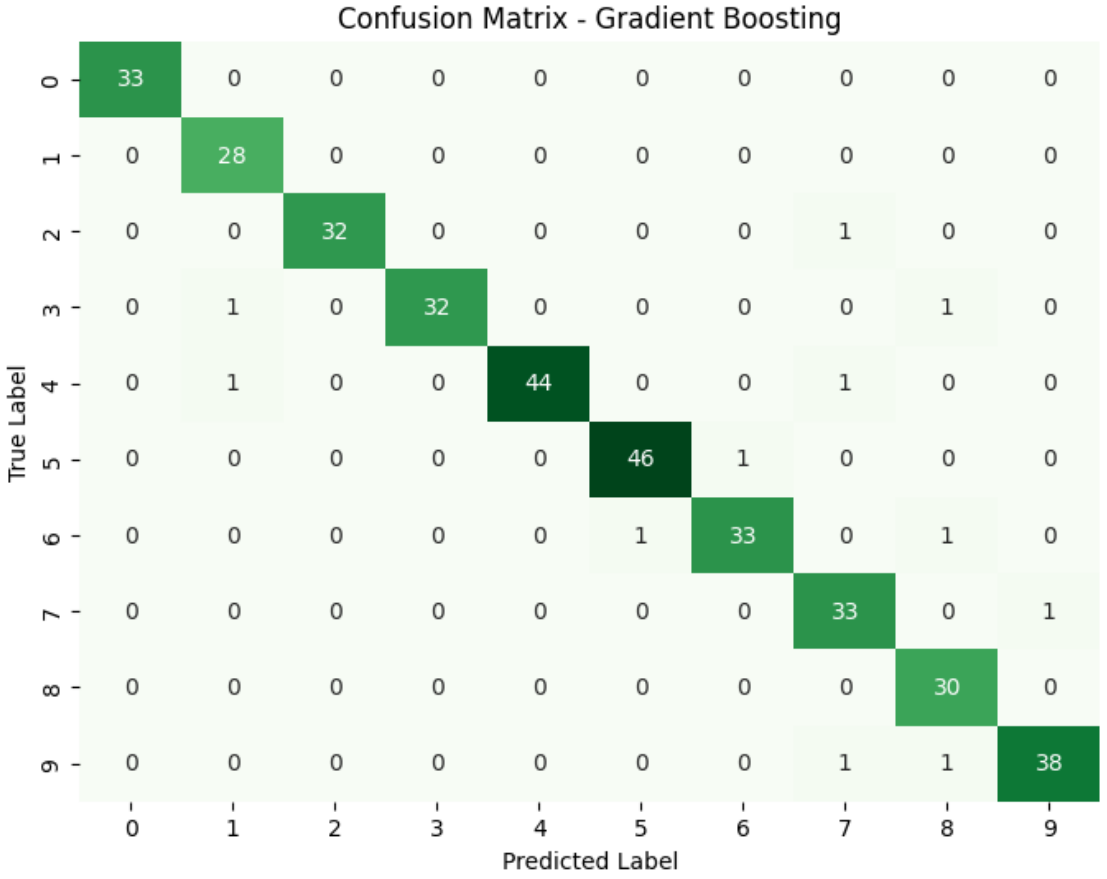
[2]:
```
# Importing necessary libraries
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Loading a sample dataset (replace with your own dataset)
digits = load_digits()
X, y = digits.data, digits.target

# Splitting 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)

# Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)

# Confusion Matrix for Random Forest
rf_conf_matrix = confusion_matrix(y_test, rf_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(rf_conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix - Random Forest')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# Feature Importance Plot for Random Forest
plt.figure(figsize=(10, 6))
sns.barplot(x=rf_model.feature_importances_, y=digits.feature_names,␣
 ↪palette='viridis')
plt.title('Feature Importance - Random Forest')
plt.xlabel('Importance Score')
```

```
plt.ylabel('Features')
plt.show()

# Gradient Boosting
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)
gb_predictions = gb_model.predict(X_test)
gb_accuracy = accuracy_score(y_test, gb_predictions)

# Confusion Matrix for Gradient Boosting
gb_conf_matrix = confusion_matrix(y_test, gb_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(gb_conf_matrix, annot=True, fmt='d', cmap='Greens', cbar=False)
plt.title('Confusion Matrix - Gradient Boosting')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# Feature Importance Plot for Gradient Boosting
plt.figure(figsize=(10, 6))
sns.barplot(x=gb_model.feature_importances_, y=digits.feature_names,␣
 ↪palette='viridis')
plt.title('Feature Importance - Gradient Boosting')
plt.xlabel('Importance Score')
plt.ylabel('Features')
plt.show()

print(f'Random Forest Accuracy: {rf_accuracy}')
print(f'Gradient Boosting Accuracy: {gb_accuracy}')
```

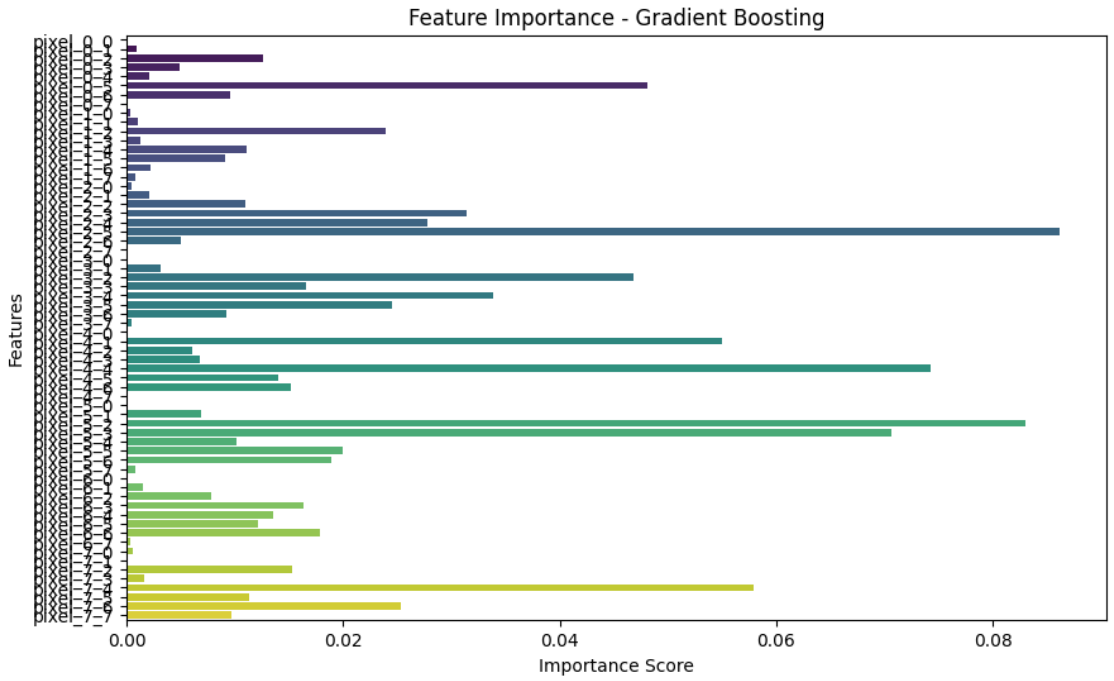



```
<ipython-input-2-cec512d604db>:33: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=rf_model.feature_importances_, y=digits.feature_names,
palette='viridis')
```


Feature Importance - Random Forest
Features
Importance Score
0.00
0.01
0.02
0.03
0.04
0.05


Confusion Matrix - Gradient Boosting
True Label
Predicted Label
0 33 0 0 0 0 0 0 0 0 0
1 0 28 0 0 0 0 0 0 0 0
2 0 0 32 0 0 0 0 1 0 0
3 0 1 0 32 0 0 0 0 1 0
4 0 1 0 0 44 0 0 1 0 0
5 0 0 0 0 0 46 1 0 0 0
6 0 0 0 0 0 1 33 0 1 0
7 0 0 0 0 0 0 0 33 0 1
8 0 0 0 0 0 0 0 0 30 0
9 0 0 0 0 0 0 0 1 1 38
0 1 2 3 4 5 6 7 8 9

```
<ipython-input-2-cec512d604db>:56: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=gb_model.feature_importances_, y=digits.feature_names,
palette='viridis')
```




```
Random Forest Accuracy: 0.9722222222222222
Gradient Boosting Accuracy: 0.9694444444444444
```

[ ]: