Phase 3 Project: Telecom Churn Classification

Edwin Joshua Kiuma
Moringa School
July 2025

Problem Statement

01 Telecom companies lose money when customers leave.

02 Goal: Predict if a customer is likely to churn.

03 Use historical data to guide retention efforts.

Business Problem

- Churn = when a customer leaves the telecom service.
- Goal: Predict churn so the business can retain customers.
- Why it matters: Acquiring new customers is more costly than keeping existing ones.

Dataset Summary

- Source: bigml_59c28831336c6604c800002a.csv
- Rows: ~3,300 customers
- Features: Usage (minutes, calls), Charges, Plans (Intl, Voicemail), etc.
- Target: Churn (0 = stay, 1 = churn)

Data Preparation

- Encoded binary columns (Yes/No → 1/0)
- Removed ID columns (Phone, State, Area Code)
- Scaled numerical features using StandardScaler

Scaled continuous variables

- Stratified train-test split (80/20)

Dataset Overview

- 3,333 records from a telecom provider.
- Target: churn (Yes/No)
- Features include call minutes, charges, plans.

Baseline Model: Logistic Regression

- Simple, interpretable model.
- Accuracy: ~86%
- Recall: ~0.38 — missed many churners
- F1 Score: ~0.52
- Confusion Matrix showed high false negatives.

Baseline Model: Decision Tree

Accuracy: 91%

F1 (Churn): 0.66

Tuned Model: Random Forest

- Used GridSearchCV to tune hyperparameters
- Best Params: n_estimators=200, max_depth=10, min_samples_split=2
- Accuracy: ~94%
- Recall: ~0.76 — major improvement in catching churners
- F1 Score: ~0.77

Tuned Model: Gradient Boosting

- Accuracy: 94%
- F1 (Churn): 0.77
- Recall: 0.71

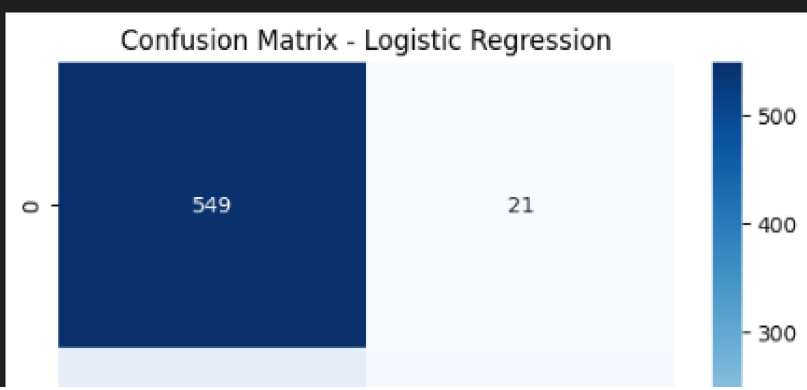
Model Comparison

Model	Precision	Recall	F1
Logistic Reg.	0.53	0.25	0.34
Decision Tree	0.69	0.63	0.66
Random Forest	**0.90**	0.68	**0.78**
Grad. Boosting	0.83	**0.71**	0.77

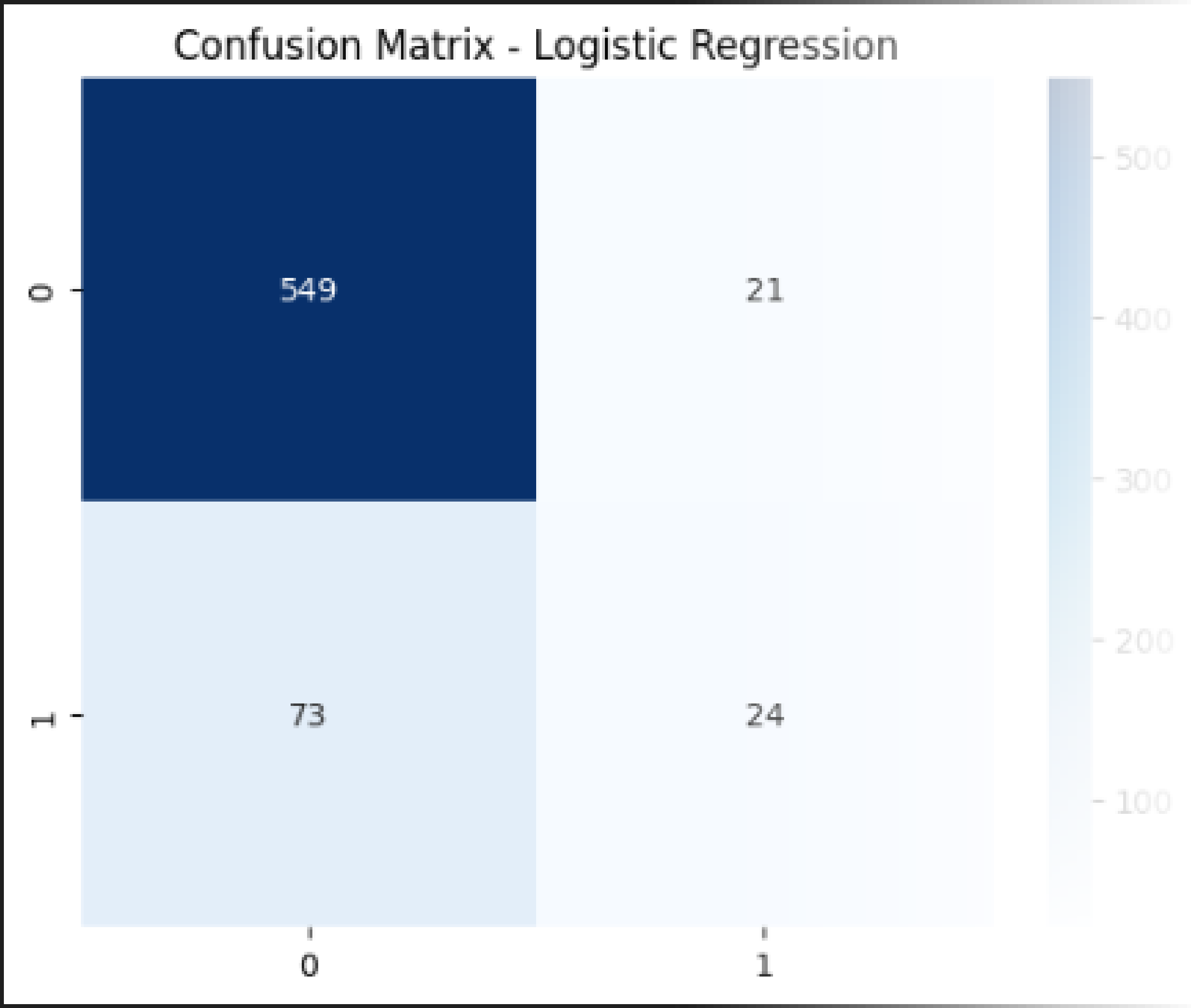
- We tested four models: Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting. While the baseline Logistic model had decent accuracy, it failed to capture churners well—its recall was only 25%. The Decision Tree improved this, but Random Forest performed best overall, with 94% accuracy and an F1 score of 0.78 for churn. Gradient Boosting was close, offering slightly better recall, but Random Forest had higher precision, making it the safest choice for minimizing false churn predictions.

```python
# ⚙️ Baseline Model: Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

logreg = Log Loading... ssion(max_iter=1000)
logreg.fit(X_train_scaled, y_train)
y_pred_lr = logreg.predict(X_test_scaled)

print("Logistic Regression:")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Recall:", recall_score(y_test, y_pred_lr))
print("F1 Score:", f1_score(y_test, y_pred_lr))

sns.heatmap(confusion_matrix(y_test, y_pred_lr), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```

```
Logistic Regression:
Accuracy: 0.8590704647676162
Recall: 0.24742268041237114
F1 Score: 0.3380281690140845
```



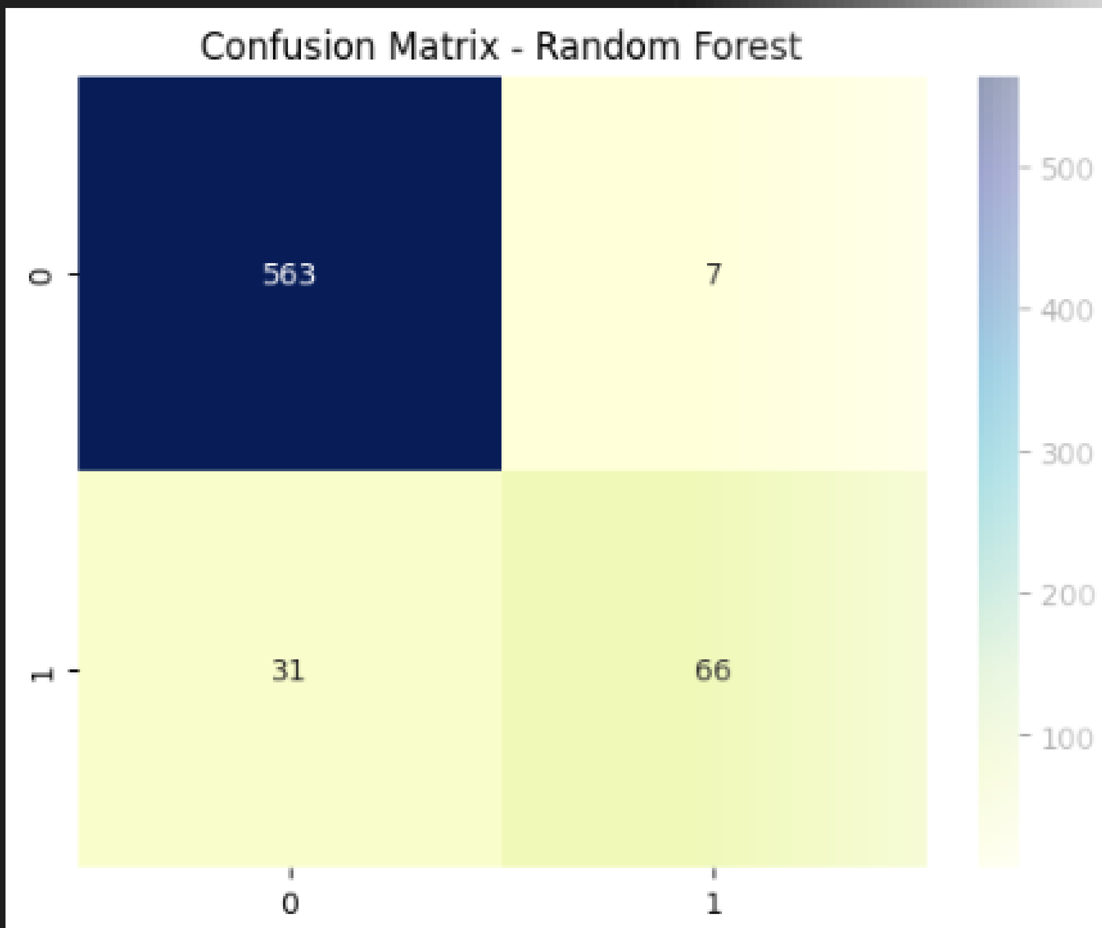
```
Logistic Regression:
Accuracy: 0.8590704647676162
Recall: 0.24742268041237114
F1 Score: 0.3380281690140845
```



```
    plt.title("Confusion Matrix - Random Forest")
    plt.show()
```

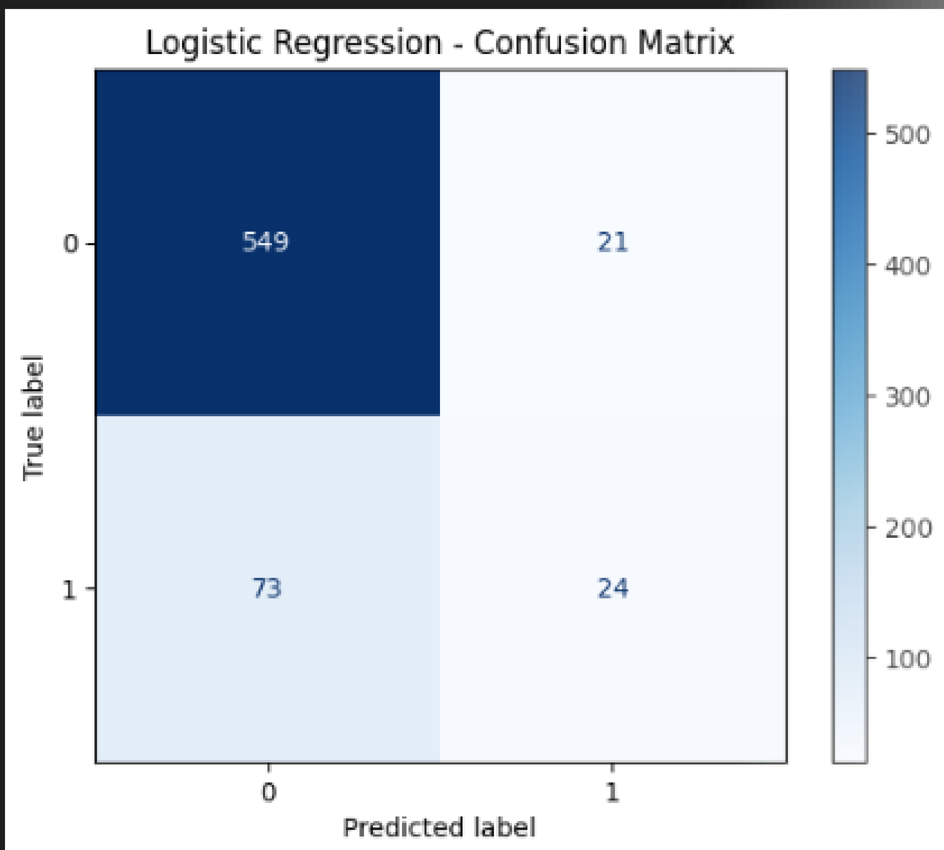
[62]

```
Random Forest (Tuned):
Best Params: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 200}
Accuracy: 0.9430284857571214
Recall: 0.6804123711340206
F1 Score: 0.7764705882352941
```



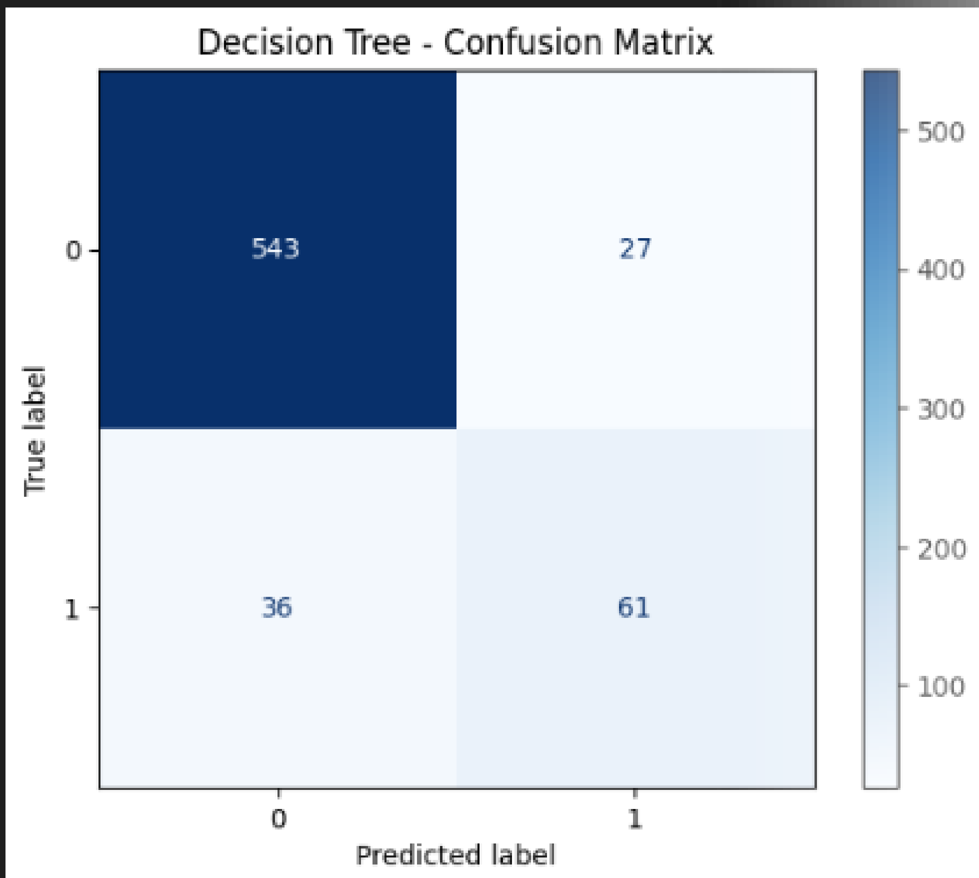
📌 Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.88	0.96	0.92	570
1	0.53	0.25	0.34	97
accuracy			0.86	667
macro avg	0.71	0.61	0.63	667
weighted avg	0.83	0.86	0.84	667



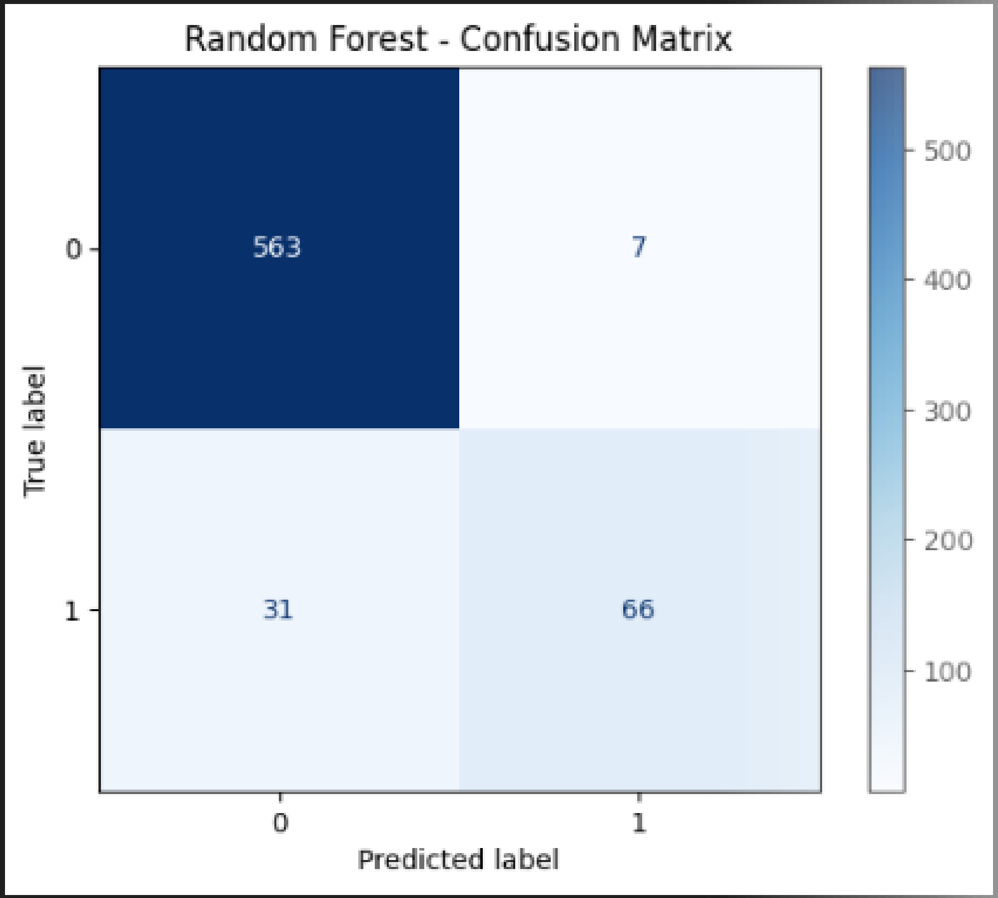
📌 Decision Tree Classification Report:

	precision	recall	f1-score	support
0	0.94	0.95	0.95	570
1	0.69	0.63	0.66	97
accuracy			0.91	667
macro avg	0.82	0.79	0.80	667
weighted avg	0.90	0.91	0.90	667



📌 Random Forest Classification Report:

	precision	recall	f1-score	support
0	0.95	0.99	0.97	570
1	0.90	0.68	0.78	97
accuracy			0.94	667
macro avg	0.93	0.83	0.87	667
weighted avg	0.94	0.94	0.94	667

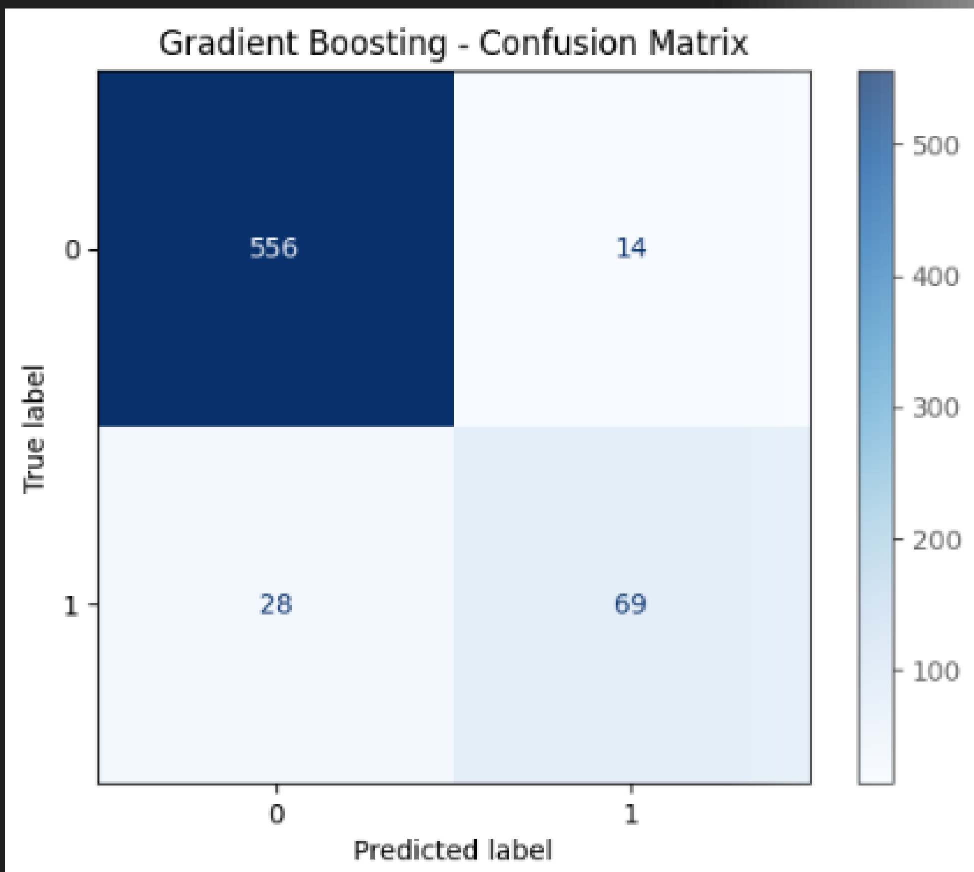


...

📌 Gradient Boosting Classification Report:

	precision	recall	f1-score	support
0	0.95	0.98	0.96	570
1	0.83	0.71	0.77	97
accuracy			0.94	667
macro avg	0.89	0.84	0.87	667
weighted avg	0.93	0.94	0.93	667

...



Model Performance Summary – Class 1: Churned Customers

Model	**Accuracy**	**Precision (1)**	**Recall (1)**	**F1-Score (1)**
Logistic Regression	0.86	0.53	0.25	0.34
Decision Tree	0.91	0.69	0.63	0.66
Random Forest	0.94	**0.90**	0.68	**0.78**
Gradient Boosting	0.94	0.83	**0.71**	0.77

Recommended Model: Random Forest

- Achieves the **highest precision** (0.90), effectively reducing false positives.
- Delivers a **balanced F1-Score (0.78)** and **strong accuracy (0.94)**.
- **Slightly lower recall** than Gradient Boosting but better overall reliability.

Use Random Forest for final deployment.
Consider Gradient Boosting when higher recall is more critical (e.g., customer retention campaigns).

Confusion Matrix & Insights

- RANDOM FOREST REDUCED FALSE NEGATIVES SIGNIFICANTLY.
- BETTER IDENTIFICATION OF HIGH-RISK CUSTOMERS.
- BUSINESS CAN NOW FOCUS ON LIKELY CHURNERS BEFORE THEY LEAVE.

Conclusion

- Final Model: Random Forest
- High precision: fewer false positives
- Supports proactive customer retention

Future Improvements

- Try SMOTE or class weights
- Explore SHAP for model explainability
- Deploy with Flask or Streamlit

Recommendations

- Prioritize outreach to customers flagged as high churn risk.
- Use model to monitor customer behavior regularly.
- Offer tailored incentives (e.g., better plans) to retain users.
- Future work: Add more features (e.g., customer support logs).

Q&A

Thank you! Any questions?