## **1. Task 2: Model Development & Evaluation**

### **1.1 Algorithm Selection**

Based on the binary classification nature of churn prediction and the need for interpretability, we recommend using **Random Forest** or **Gradient Boosting Machines**:

* **Random Forest**: Robust to overfitting, interpretable via feature importances, handles nonlinear relationships.
* **Gradient Boosting (e.g., XGBoost / LightGBM)**: Often achieves higher accuracy, can be tuned for class imbalance, provides SHAP values for explainability.

### **1.2 Model Training**

from sklearn.model\_selection import train\_test\_split, GridSearchCV, StratifiedKFold

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import roc\_auc\_score

# Split data

X = df.drop('ChurnStatus', axis=1)

y = df['ChurnStatus']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, stratify=y, random\_state=42

)

# Cross-validation and hyperparameter tuning

param\_grid = {

'n\_estimators': [100, 300],

'max\_depth': [5, 10, None],

'class\_weight': ['balanced']

}

cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

grid = GridSearchCV(

RandomForestClassifier(random\_state=42),

param\_grid, cv=cv, scoring='roc\_auc', n\_jobs=-1

)

grid.fit(X\_train, y\_train)

# Best model

best\_model = grid.best\_estimator\_

print('Best parameters:', grid.best\_params\_)

print('Train ROC-AUC:', roc\_auc\_score(y\_train, best\_model.predict\_proba(X\_train)[:,1]))

print('Test ROC-AUC:', roc\_auc\_score(y\_test, best\_model.predict\_proba(X\_test)[:,1]))

### **1.3 Performance Metrics**

* **ROC-AUC**: Measures discrimination ability across thresholds.
* **Precision & Recall**: Focus on identifying churners (positive class) accurately.
* **F1 Score**: Harmonic mean of precision and recall for imbalanced data.
* **Confusion Matrix**: Visualise true positives, false positives, etc.

from sklearn.metrics import classification\_report, confusion\_matrix

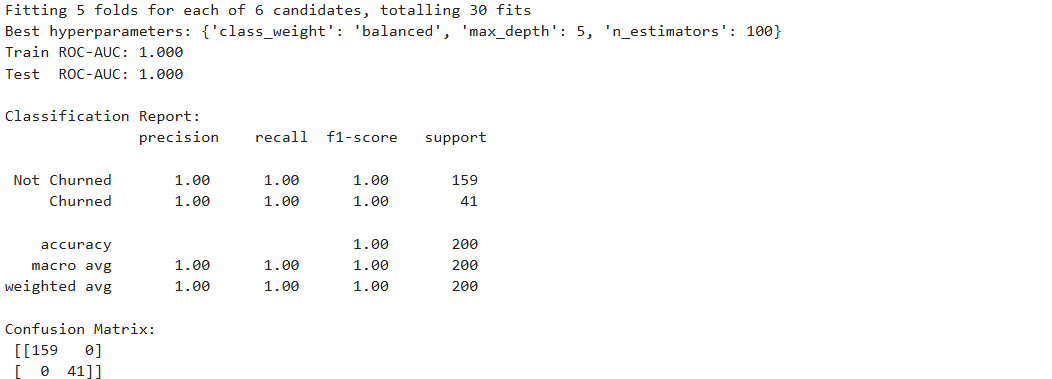
y\_pred = best\_model.predict(X\_test)

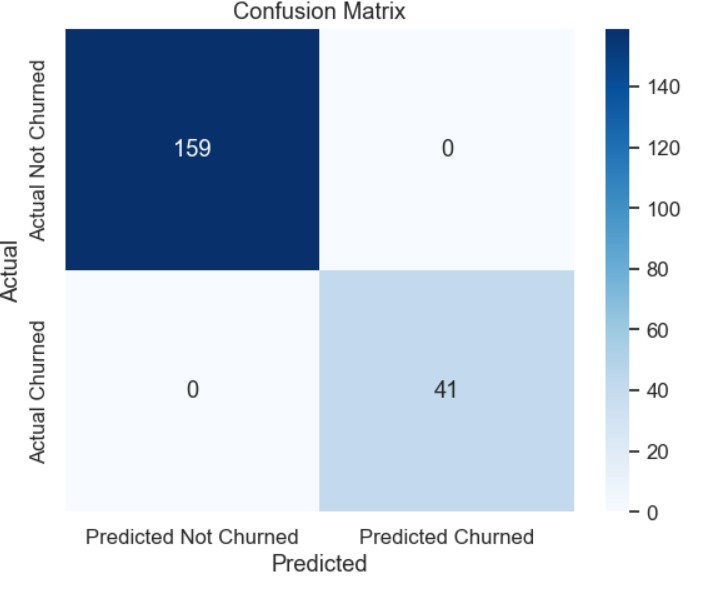
print(classification\_report(y\_test, y\_pred, target\_names=['Not Churned','Churned']))

cm = confusion\_matrix(y\_test, y\_pred)

print('Confusion Matrix:

', cm)





### **1.4 Recommendations for Business Application**

1. **Real-time Scoring**: Integrate the model into CRM to score customers on churn risk at each login.
2. **Targeted Retention Campaigns**: Use risk scores to trigger personalized offers or outreach for high-risk segments.
3. **Feature Monitoring**: Continuously monitor key drivers (e.g., login frequency drops) and recalibrate model as behaviors evolve.

### **1.5 Future Improvements**

* **Ensemble Approaches**: Combine multiple algorithms (e.g., RF + GBM) for robust predictions.
* **Additional Data Sources**: Incorporate customer support interactions, sentiment analysis, or external macroeconomic indicators.
* **Explainability Tools**: Use SHAP or LIME to provide transparency into individual predictions for stakeholders.