Random Forest Classifier model 1

January 12, 2026

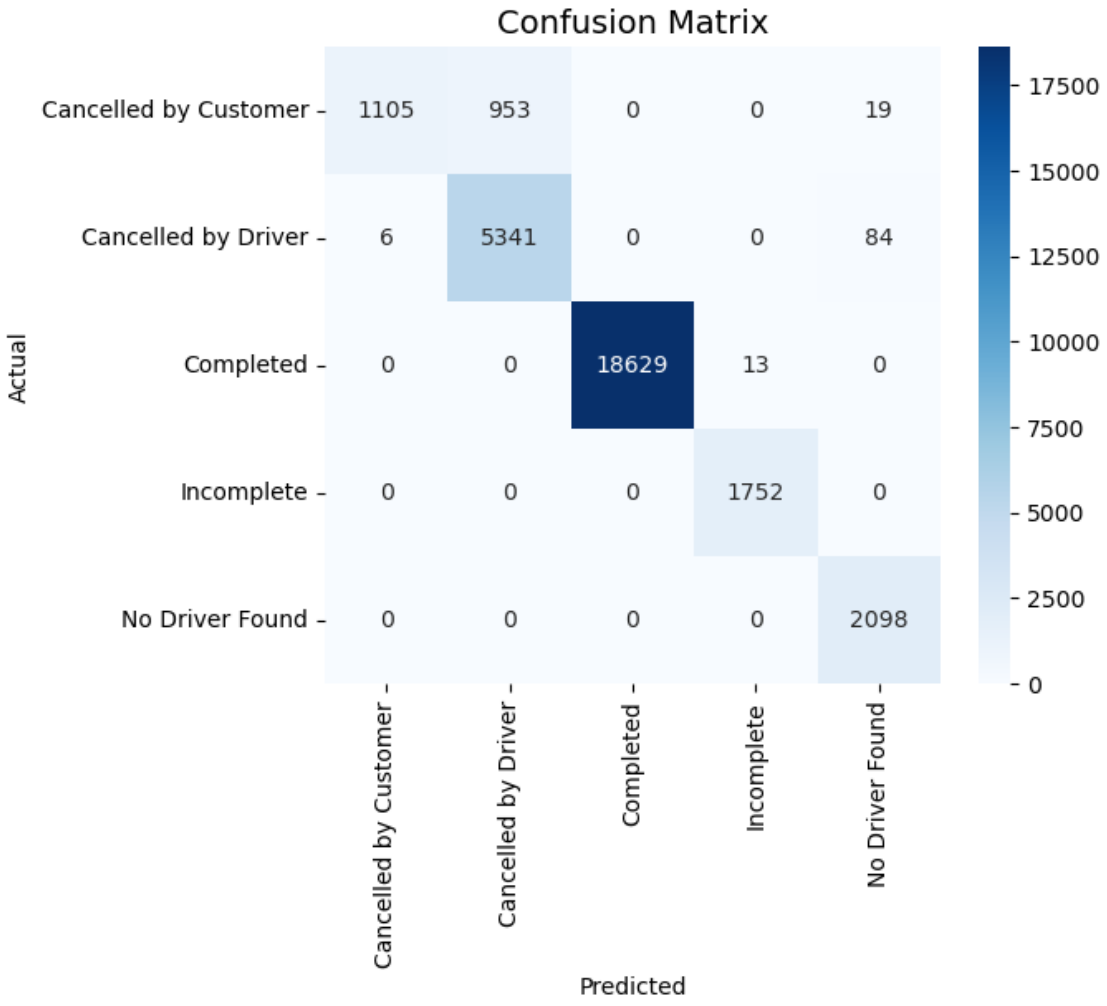
[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, label_binarize
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    accuracy_score, f1_score, classification_report, confusion_matrix,
    roc_curve, auc, roc_auc_score
)
from sklearn.ensemble import RandomForestClassifier
from itertools import cycle

# Load dataset
df = pd.read_csv("ncr_ride_bookings.csv")

# Convert Date and Time
df['Date'] = pd.to_datetime(df['Date'])
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day

df['Time'] = pd.to_timedelta(df['Time'])
df['seconds'] = df['Time'].dt.total_seconds().astype(int)

# Select relevant columns
df = df[['year', 'month', 'day', 'seconds', 'Booking Status', 'Vehicle Type',
         'Pickup Location', 'Drop Location', 'Avg VTAT', 'Avg CTAT',
         'Booking Value', 'Ride Distance', 'Driver Ratings', 'Customer Rating',
         'Payment Method']]

# Handle missing values
df.fillna(df.median(numeric_only=True), inplace=True)

# Encode categorical variables
x_categorical = df.select_dtypes(include=['object']).apply(lambda col:␣
 ↪LabelEncoder().fit_transform(col))
```

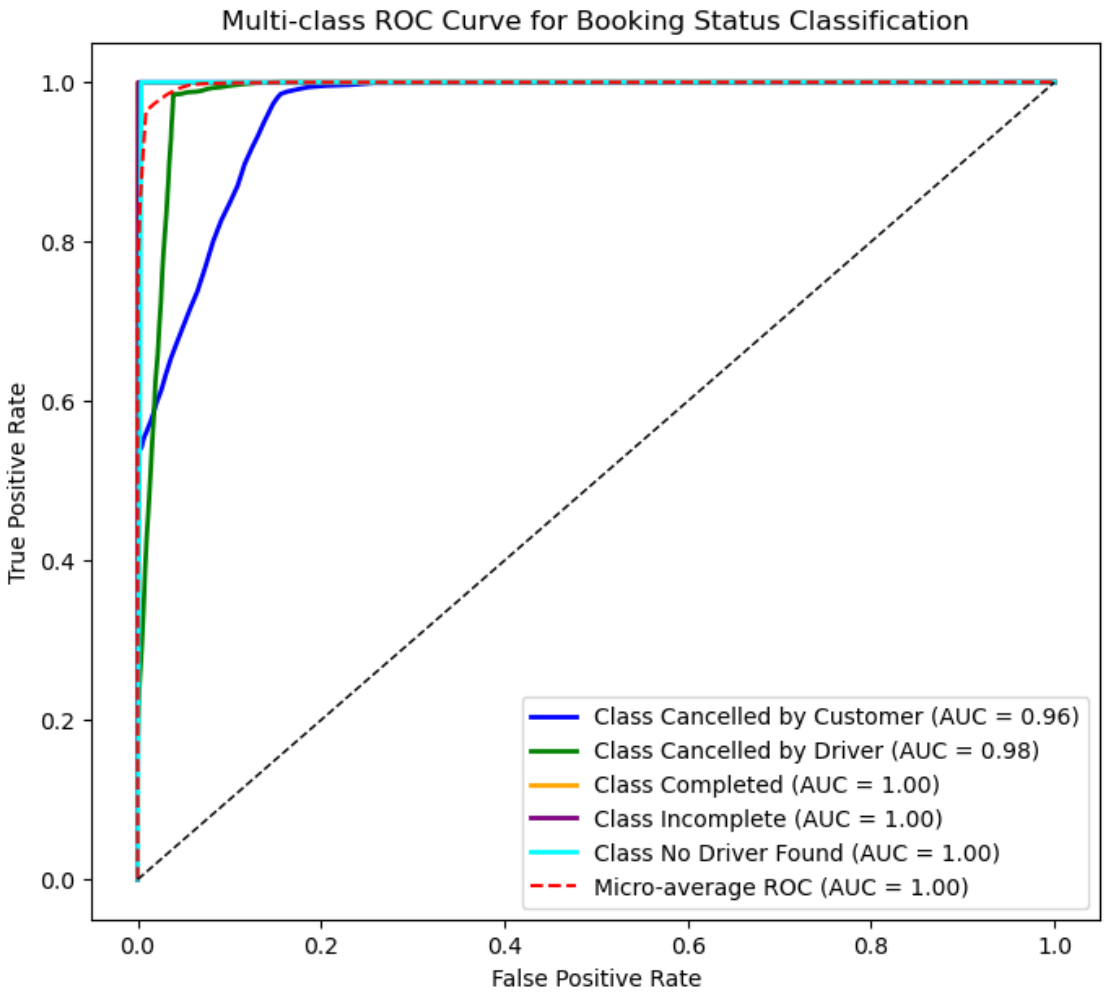
```
x_numerical = df.select_dtypes(exclude=['object'])
X = pd.concat([x_numerical, x_categorical], axis=1).drop(columns=['Booking␣
 ↪Status'])

# Encode target
le = LabelEncoder()
y = le.fit_transform(df['Booking Status'])

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

# Random Forest Classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=0)
classifier.fit(X_train, y_train)

# Predictions
y_pred = classifier.predict(X_test)
y_prob = classifier.predict_proba(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score (weighted): {f1:.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred,␣
 ↪target_names=le.classes_))

# ========================
# Confusion Matrix
# ========================
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=le.classes_, yticklabels=le.classes_)
plt.title("Confusion Matrix", fontsize=14)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ========================
# ROC Curve (binary & multi-class)
# ========================
n_classes = len(le.classes_)

if n_classes == 2:
```

```
    # Binary ROC
    fpr, tpr, _ = roc_curve(y_test, y_prob[:, 1])
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(7,6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_auc:.
↪3f})')
    plt.plot([0, 1], [0, 1], color='red', linestyle='--')
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ROC Curve for Booking Status Classification")
    plt.legend(loc="lower right")
    plt.show()

else:
    # Multi-class ROC using One-vs-Rest
    y_test_bin = label_binarize(y_test, classes=np.arange(n_classes))

    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_prob[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    # Micro-average ROC
    fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), y_prob.
↪ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

    # Plot ROC curves
    plt.figure(figsize=(8,7))
    colors = cycle(['blue', 'green', 'orange', 'purple', 'cyan', 'brown'])
    for i, color in zip(range(n_classes), colors):
        plt.plot(fpr[i], tpr[i], color=color, lw=2,
                 label=f'Class {le.classes_[i]} (AUC = {roc_auc[i]:.2f})')

    plt.plot(fpr["micro"], tpr["micro"], color='red', linestyle='--',
             label=f'Micro-average ROC (AUC = {roc_auc["micro"]:.2f})')

    plt.plot([0, 1], [0, 1], 'k--', lw=1)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Multi-class ROC Curve for Booking Status Classification")
    plt.legend(loc="lower right")
    plt.show()
```

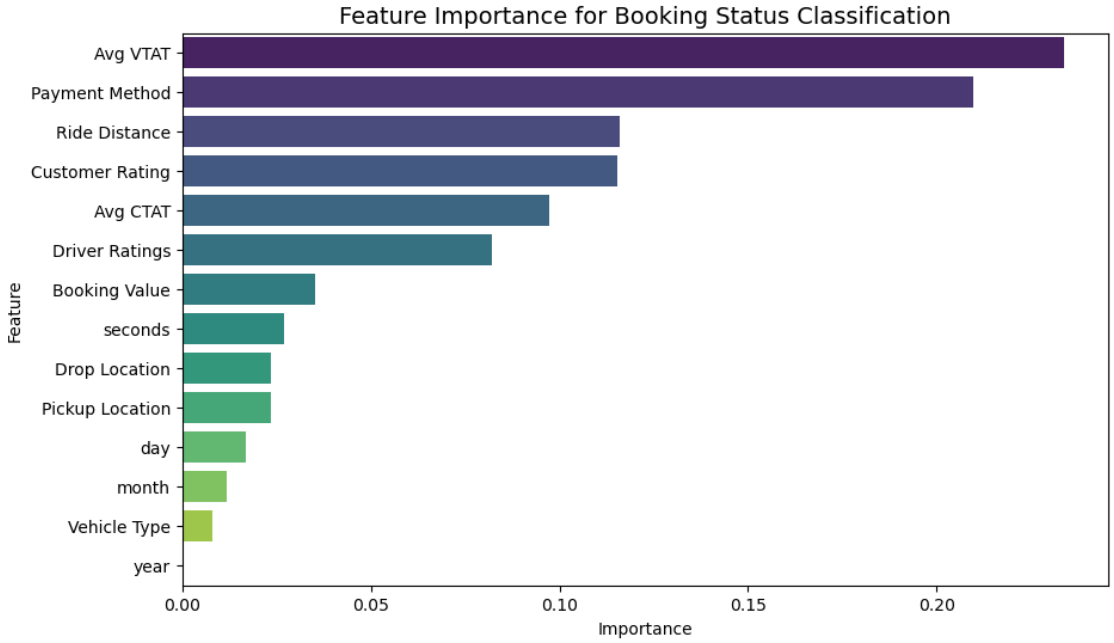
```python
# ========================
# Feature Importance
# ========================
importances = classifier.feature_importances_
features = X.columns

feat_imp = pd.DataFrame({'Feature': features, 'Importance': importances})
feat_imp = feat_imp.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x='Importance', y='Feature', data=feat_imp, palette='viridis')
plt.title("Feature Importance for Booking Status Classification", fontsize=14)
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```

```
Accuracy: 0.9642
F1 Score (weighted): 0.9605

Classification Report:
                      precision    recall  f1-score   support

Cancelled by Customer       0.99      0.53      0.69      2077
  Cancelled by Driver       0.85      0.98      0.91      5431
            Completed       1.00      1.00      1.00     18642
           Incomplete       0.99      1.00      1.00      1752
      No Driver Found       0.95      1.00      0.98      2098

             accuracy                           0.96     30000
            macro avg       0.96      0.90      0.92     30000
         weighted avg       0.97      0.96      0.96     30000
```


Confusion Matrix
Cancelled by Customer
Cancelled by Driver
Completed
Incomplete
No Driver Found
Actual
1105
953
0
0
19
6
5341
0
0
84
0
0
18629
13
0
0
0
0
1752
0
0
0
0
0
2098
Cancelled by Customer
Cancelled by Driver
Completed
Incomplete
No Driver Found
Predicted
17500
15000
12500
10000
7500
5000
2500
0




```
C:\Users\user\AppData\Local\Temp\ipykernel_19536\2298432815.py:137:
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='Importance', y='Feature', data=feat_imp, palette='viridis')
```


Feature Importance for Booking Status Classification
Avg VTAT
Payment Method
Ride Distance
Customer Rating
Avg CTAT
Driver Ratings
Booking Value
seconds
Drop Location
Pickup Location
day
month
Vehicle Type
year
Feature
0.00
0.05
0.10
0.15
0.20
Importance

[ ]: