```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# Provide the path to your dataset in Google Drive
file_path = '/content/drive/MyDrive/booking.csv'

# Load the dataset
hotel_data = pd.read_csv(file_path)
print("Data Preview:\n", hotel_data.head())

# Data Exploration: Check for missing values and dataset overview
print("Missing Values Summary:\n", hotel_data.isnull().sum())
print("Dataset shape:", hotel_data.shape)

# Feature Engineering
# Convert 'date of reservation' to datetime format and extract useful features
hotel_data['date of reservation'] = pd.to_datetime(hotel_data['date of reservation'], errors='coerce')
hotel_data['reservation_month'] = hotel_data['date of reservation'].dt.month
hotel_data['reservation_year'] = hotel_data['date of reservation'].dt.year

# Convert 'booking status' to a binary target variable
hotel_data['cancellation_status'] = hotel_data['booking status'].apply(lambda x: 1 if x == 'Canceled' else 0)

# Encode categorical variables using one-hot encoding
hotel_data_encoded = pd.get_dummies(hotel_data, columns=['type of meal', 'room type', 'market segment type'], drop_first=True)

# Drop unnecessary columns and handle missing values
hotel_data_encoded.drop(columns=['date of reservation', 'booking status'], inplace=True)
hotel_data_encoded = hotel_data_encoded.dropna()

# Separate features (X) and target (Y)
X = hotel_data_encoded.drop(['cancellation_status', 'Booking_ID'], axis=1)
Y = hotel_data_encoded['cancellation_status']

# Split 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)

# Initialize and train Random Forest model
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)

# Predict on the test set
random_forest_y_pred = random_forest_model.predict(X_test)

# Evaluate the model
random_forest_metrics = {
    "Accuracy": accuracy_score(y_test, random_forest_y_pred),
    "Precision": precision_score(y_test, random_forest_y_pred),
    "Recall": recall_score(y_test, random_forest_y_pred),
    "F1 Score": f1_score(y_test, random_forest_y_pred),
}
print("Random Forest Metrics:", random_forest_metrics)

# Confusion Matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, random_forest_y_pred))

# Feature Importance
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': random_forest_model.feature_importances_
}).sort_values(by='Importance', ascending=False)
print("Feature Importances:\n", feature_importances)

# Cross-validation for robust evaluation
cv_scores = cross_val_score(random_forest_model, X_train, y_train, cv=5, scoring='f1')
print("Cross-validated F1 scores:", cv_scores)
print("Average F1 score:", cv_scores.mean())

# Hyperparameter Tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
```

```
    'max_features': ['sqrt', 'log2']
}

grid_search_rf = GridSearchCV(
    RandomForestClassifier(random_state=42),
    param_grid,
    scoring='f1',
    cv=3,
    verbose=1
)
grid_search_rf.fit(X_train, y_train)

# Get the best model and evaluate
best_rf_model = grid_search_rf.best_estimator_
best_rf_y_pred = best_rf_model.predict(X_test)

optimized_rf_metrics = {
    "Accuracy": accuracy_score(y_test, best_rf_y_pred),
    "Precision": precision_score(y_test, best_rf_y_pred),
    "Recall": recall_score(y_test, best_rf_y_pred),
    "F1 Score": f1_score(y_test, best_rf_y_pred),
}
print("Optimized Random Forest Metrics:", optimized_rf_metrics)

# Final Cross-validation for the best model
cv_scores_optimized = cross_val_score(best_rf_model, X_train, y_train, cv=5, scoring='f1')
print("Cross-validated F1 scores (optimized):", cv_scores_optimized)
print("Average F1 score (optimized):", cv_scores_optimized.mean())
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Data Preview:
   Booking_ID  number of adults  number of children  number of weekend nights  \
0   INN00001                 1                   1                         2
1   INN00002                 1                   0                         1
2   INN00003                 2                   1                         1
3   INN00004                 1                   0                         0
4   INN00005                 1                   0                         1

   number of week nights  type of meal  car parking space    room type  \
0                      5   Meal Plan 1                  0  Room_Type 1
1                      3  Not Selected                  0  Room_Type 1
2                      3   Meal Plan 1                  0  Room_Type 1
3                      2   Meal Plan 1                  0  Room_Type 1
4                      2  Not Selected                  0  Room_Type 1

   lead time market segment type  repeated  P-C  P-not-C  average price  \
0        224             Offline         0    0        0          88.00
1          5              Online         0    0        0         106.68
2          1              Online         0    0        0          50.00
3        211              Online         0    0        0         100.00
4         48              Online         0    0        0          77.00

   special requests date of reservation booking status
0                 0           10/2/2015   Not_Canceled
1                 1           11/6/2018   Not_Canceled
2                 0           2/28/2018       Canceled
3                 1           5/20/2017       Canceled
4                 0           4/11/2018       Canceled
Missing Values Summary:
 Booking_ID                  0
number of adults             0
number of children           0
number of weekend nights     0
number of week nights        0
type of meal                 0
car parking space            0
room type                    0
lead time                    0
market segment type          0
repeated                     0
P-C                          0
P-not-C                      0
average price                0
special requests             0
date of reservation          0
booking status               0
dtype: int64
Dataset shape: (36285, 17)
Random Forest Metrics: {'Accuracy': 0.8971034482758621, 'Precision': 0.8647191011235955, 'Recall': 0.8121570282819756, 'F1 Score'
Confusion Matrix:
 [[4580  301]
 [ 445 1924]]
Feature Importances:
                            Feature  Importance
5                         lead time    0.357203
9                      average price    0.184194
```