PYTHON PANDA RPOGRAM FOR HOTEL BOOKING ANALYSIS

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Load the dataset
FILEPATH = 'hotel_bookings.xlsx'
df = pd.read_excel(FILEPATH)

# Preprocessing
# Fill missing values
for column in ['country', 'agent', 'company']:
    df[column].fillna('None', inplace=True)

# Convert non-numeric columns to numeric
le = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = le.fit_transform(df[column])

# Define the target variable and features
X = df.drop('is_canceled', axis=1)
```

```
y = df['is_canceled']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict and evaluate
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)

# Feature importance
feature_importances = pd.DataFrame(model.feature_importances_, index = X_train.columns, columns=['importance']).sort_values('importance', ascending=False)

print('Model Accuracy:', accuracy)
print(feature_importances.head(10))
```

```
# Convert all object columns to strings to ensure uniform data types
for column in df.select_dtypes(include=['object']).columns:
    df[column] = df[column].astype(str)

# Re-apply LabelEncoder
le = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = le.fit_transform(df[column])

# Continue with the previous steps
# Define the target variable and features
X = df.drop('is_canceled', axis=1)
y = df['is_canceled']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict and evaluate
```

```
predictions = model.predict(X_test)

accuracy = accuracy_score(y_test, predictions)


# Feature importance

feature_importances = pd.DataFrame(model.feature_importances_,
index = X_train.columns,
columns=['importance']).sort_values('importance', ascending=False)


print('Model Accuracy:', accuracy)

print(feature_importances.head(10))
```