```python
import pandas as pd
import numpy as np

# Load
data = pd.read_csv("customer_booking.csv", encoding='latin1')
data.head()
```

…	num_pa…	sales_…	t..	purch…	length_o…	flig…	fl…	…	bool
0	2	Internet	RoundTrip	262	19	7	Sat	AKLDEL	New
1	1	Internet	RoundTrip	112	20	3	Sat	AKLDEL	New
2	2	Internet	RoundTrip	243	22	17	Wed	AKLDEL	Indi
3	1	Internet	RoundTrip	96	31	4	Sat	AKLDEL	New
4	2	Internet	RoundTrip	68	22	15	Wed	AKLDEL	Indi

Rows: 5 Expand

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Set target variable
target_col = 'booking_complete'

# Encode categorical variables
df_encoded = data.copy()
for col in df_encoded.select_dtypes(include=['object', 'category']).columns:
    df_encoded[col] = LabelEncoder().fit_transform(df_encoded[col].astype(str))

# Handle missing values (simple fill for demonstration)
df_encoded = df_encoded.fillna(df_encoded.median(numeric_only=True))

# Split features and target
X = df_encoded.drop(columns=[target_col])
y = df_encoded[target_col]

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

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score, roc_curve

# Train Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
rf_probs = rf.predict_proba(X_test)[:,1]
rf_auc = roc_auc_score(y_test, rf_probs)

# Train Logistic Regression
lr = LogisticRegression(max_iter=1000, random_state=42)
lr.fit(X_train, y_train)
lr_probs = lr.predict_proba(X_test)[:,1]
lr_auc = roc_auc_score(y_test, lr_probs)

# Train XGBoost
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb.fit(X_train, y_train)
xgb_probs = xgb.predict_proba(X_test)[:,1]
xgb_auc = roc_auc_score(y_test, xgb_probs)

print(f"Random Forest ROC-AUC: {rf_auc:.3f}")
print(f"Logistic Regression ROC-AUC: {lr_auc:.3f}")
print(f"XGBoost ROC-AUC: {xgb_auc:.3f}")
```

```
Random Forest ROC-AUC: 0.776
Logistic Regression ROC-AUC: 0.658
XGBoost ROC-AUC: 0.783
```

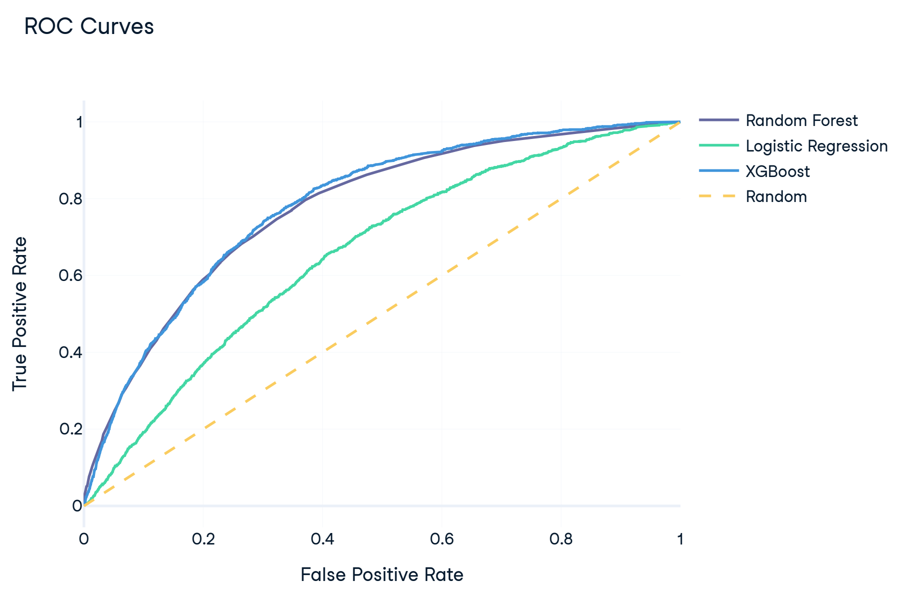
```
print(df_encoded.corr()[target_col].sort_values(ascending=False))
```

```
booking_complete         1.000000
booking_origin           0.130804
wants_extra_baggage      0.068139
wants_preferred_seat     0.050116
trip_type                0.027021
wants_in_flight_meals    0.026511
num_passengers           0.024116
flight_day               0.010929
flight_hour              0.007127
route                   -0.008488
purchase_lead           -0.022131
sales_channel           -0.041060
length_of_stay          -0.042408
flight_duration         -0.106266
Name: booking_complete, dtype: float64
```

```
import plotly.graph_objs as go

# Plot ROC curves
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf_probs)
fpr_lr, tpr_lr, _ = roc_curve(y_test, lr_probs)
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, xgb_probs)

fig = go.Figure()
fig.add_trace(go.Scatter(x=fpr_rf, y=tpr_rf, mode='lines', name='Random Forest'))
fig.add_trace(go.Scatter(x=fpr_lr, y=tpr_lr, mode='lines', name='Logistic Regression'))
fig.add_trace(go.Scatter(x=fpr_xgb, y=tpr_xgb, mode='lines', name='XGBoost'))
fig.add_trace(go.Scatter(x=[0,1], y=[0,1], mode='lines', name='Random', line=dict(dash='dash')))
fig.update_layout(title='ROC Curves', xaxis_title='False Positive Rate', yaxis_title='True Positive Rate', width=700,
height=500)
fig.show()
```



can you explain the graph. my graph in cell above

The graph displayed above is a set of ROC (Receiver Operating Characteristic) curves for three different classification models: Random Forest, Logistic Regression, and XGBoost. Here is an explanation of the key elements:

- **ROC Curve:** This curve plots the True Positive Rate (TPR, or sensitivity) against the False Positive Rate (FPR, or 1-specificity) at various threshold settings. It is used to evaluate the performance of a binary classifier.
- **Curves for Each Model:** Each line represents the performance of a different model. The closer a curve follows the left-hand border and then the top border of the ROC space, the better the model's performance.
- **Random Line:** The diagonal dashed line represents the performance of a random classifier (i.e., no predictive power). Any model performing above this line is better than random guessing.
- **Interpretation:** The area under each curve (AUC) is a common summary metric. A higher AUC indicates a better performing model. If one model's curve is consistently above the others, it generally means that model is better at distinguishing between the classes.

In summary, this graph allows you to visually compare the classification performance of the three models. The model with the curve closest to the top-left corner is typically the best performer.

```python
# Random Forest feature importance
import pandas as pd

rf_importances = pd.Series(rf.feature_importances_, index=X_train.columns)
print(rf_importances.sort_values(ascending=False))

# XGBoost feature importance
xgb_importances = pd.Series(xgb.feature_importances_, index=X_train.columns)
print(xgb_importances.sort_values(ascending=False))
```

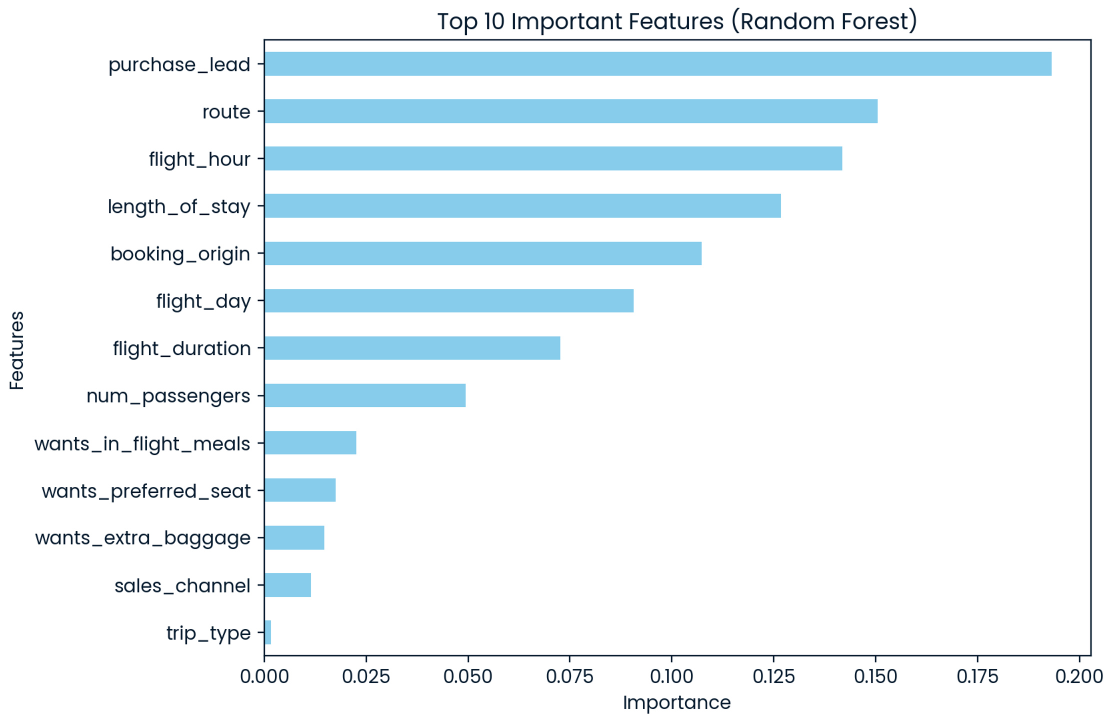
```
purchase_lead            0.193172
route                    0.150536
flight_hour              0.141817
length_of_stay           0.126805
booking_origin           0.107279
flight_day               0.090645
flight_duration          0.072579
num_passengers           0.049366
wants_in_flight_meals    0.022535
wants_preferred_seat     0.017435
wants_extra_baggage      0.014765
sales_channel            0.011428
trip_type                0.001638
dtype: float64
booking_origin           0.242937
sales_channel            0.103964
wants_extra_baggage      0.093583
route                    0.076988
wants_preferred_seat     0.075219
flight_duration          0.065109
trip_type                0.061127
length_of_stay           0.054837
wants_in_flight_meals    0.053611
purchase_lead            0.046070
num_passengers           0.043740
flight_day               0.042006
flight_hour              0.040810
dtype: float32
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Get feature importances from Random Forest
importances = pd.Series(rf.feature_importances_, index=X_train.columns)

# Sort and select top 10
top_features = importances.sort_values(ascending=False).head(20)

# Plot
plt.figure(figsize=(8,6))
top_features.plot(kind='barh', color='skyblue')
plt.xlabel("Importance")
plt.ylabel("Features")
plt.title("Top 10 Important Features (Random Forest)")
plt.gca().invert_yaxis()  # Highest importance at the top
plt.show()
```


Top 10 Important Features (Random Forest)
purchase_lead
route
flight_hour
length_of_stay
booking_origin
flight_day
flight_duration
num_passengers
wants_in_flight_meals
wants_preferred_seat
wants_extra_baggage
sales_channel
trip_type
Features
Importance