Task

Bu veri setimizde otel rezervasyonu yapan müşterilerden elde edilen veriler ile rezervasyon yapan müşterilerden ilerleyen zamanlarda rezervasyonunu iptal edip etmeyeceği tahmin etmek istenmektedirThe goal is to predict whether customers who made hotel reservations will cancel their reservations in the future using the data obtained from customers who made hotel reservations.

In [15]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LogisticRegressionCV
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve,roc_auc_score, RocCurveDisplay
```

In [16]:
```
hb = pd.read_csv("hotel_bookings.csv")
```

First, we will check the variable types and presence of null values in our dataset.

In [17]:
```
hb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   hotel                           119390 non-null  object 
 1   is_canceled                     119390 non-null  int64  
 2   lead_time                       119390 non-null  int64  
 3   arrival_date_year               119390 non-null  int64  
 4   arrival_date_month              119390 non-null  object 
 5   arrival_date_week_number        119390 non-null  int64  
 6   arrival_date_day_of_month       119390 non-null  int64  
 7   stays_in_weekend_nights         119390 non-null  int64  
 8   stays_in_week_nights            119390 non-null  int64  
 9   adults                          119390 non-null  int64  
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64  
 12  meal                            119390 non-null  object 
 13  country                         118902 non-null  object 
 14  market_segment                  119390 non-null  object 
 15  distribution_channel            119390 non-null  object 
 16  is_repeated_guest               119390 non-null  int64  
 17  previous_cancellations          119390 non-null  int64  
 18  previous_bookings_not_canceled  119390 non-null  int64  
 19  reserved_room_type              119390 non-null  object 
 20  assigned_room_type              119390 non-null  object 
 21  booking_changes                 119390 non-null  int64  
 22  deposit_type                    119390 non-null  object 
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64  
 26  customer_type                   119390 non-null  object 
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64  
 29  total_of_special_requests       119390 non-null  int64  
 30  reservation_status              119390 non-null  object 
 31  reservation_status_date         119390 non-null  object 
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

In [18]:
```
hb.isnull().sum()
```

Out[18]:
```
hotel                               0
is_canceled                         0
lead_time                           0
arrival_date_year                   0
arrival_date_month                  0
arrival_date_week_number            0
arrival_date_day_of_month           0
stays_in_weekend_nights             0
stays_in_week_nights                0
adults                              0
children                            4
babies                              0
meal                                0
country                           488
market_segment                      0
distribution_channel                0
is_repeated_guest                   0
previous_cancellations              0
previous_bookings_not_canceled      0
reserved_room_type                  0
assigned_room_type                  0
booking_changes                     0
deposit_type                        0
agent                           16340
company                        112593
days_in_waiting_list                0
customer_type                       0
adr                                 0
required_car_parking_spaces         0
total_of_special_requests           0
reservation_status                  0
reservation_status_date             0
dtype: int64
```

In [19]:
```
newhb = hb.drop(['agent','company','country'],axis=1)
newhb = newhb.dropna()
```

In [20]:
```
newhb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119386 entries, 0 to 119389
Data columns (total 29 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   hotel                           119386 non-null  object 
 1   is_canceled                     119386 non-null  int64  
 2   lead_time                       119386 non-null  int64  
 3   arrival_date_year               119386 non-null  int64  
 4   arrival_date_month              119386 non-null  object 
 5   arrival_date_week_number        119386 non-null  int64  
 6   arrival_date_day_of_month       119386 non-null  int64  
 7   stays_in_weekend_nights         119386 non-null  int64  
 8   stays_in_week_nights            119386 non-null  int64  
 9   adults                          119386 non-null  int64  
 10  children                        119386 non-null  float64
 11  babies                          119386 non-null  int64  
 12  meal                            119386 non-null  object 
 13  market_segment                  119386 non-null  object 
 14  distribution_channel            119386 non-null  object 
 15  is_repeated_guest               119386 non-null  int64  
 16  previous_cancellations          119386 non-null  int64  
 17  previous_bookings_not_canceled  119386 non-null  int64  
 18  reserved_room_type              119386 non-null  object 
 19  assigned_room_type              119386 non-null  object 
 20  booking_changes                 119386 non-null  int64  
 21  deposit_type                    119386 non-null  object 
 22  days_in_waiting_list            119386 non-null  int64  
 23  customer_type                   119386 non-null  object 
 24  adr                             119386 non-null  float64
 25  required_car_parking_spaces     119386 non-null  int64  
 26  total_of_special_requests       119386 non-null  int64  
 27  reservation_status              119386 non-null  object 
 28  reservation_status_date         119386 non-null  object 
dtypes: float64(2), int64(16), object(11)
memory usage: 27.3+ MB
```

Due to a high number of null values in the variables "Agent," "Company," and "Country," we decided to remove these columns from the dataset. Additionally, there were four null values in the "Children" variable, so instead of removing the entire variable, we opted to exclude the observations with these four null values from the dataset.

In [21]:
```
newhb = newhb.drop(["hotel","meal","arrival_date_month","market_segment","distri
```

In [22]:
```
newhb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119386 entries, 0 to 119389
Data columns (total 18 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   is_canceled                     119386 non-null  int64  
 1   lead_time                       119386 non-null  int64  
 2   arrival_date_year               119386 non-null  int64  
 3   arrival_date_week_number        119386 non-null  int64  
 4   arrival_date_day_of_month       119386 non-null  int64  
 5   stays_in_weekend_nights         119386 non-null  int64  
 6   stays_in_week_nights            119386 non-null  int64  
 7   adults                          119386 non-null  int64  
 8   children                        119386 non-null  float64
 9   babies                          119386 non-null  int64  
 10  is_repeated_guest               119386 non-null  int64  
 11  previous_cancellations          119386 non-null  int64  
 12  previous_bookings_not_canceled  119386 non-null  int64  
 13  booking_changes                 119386 non-null  int64  
 14  days_in_waiting_list            119386 non-null  int64  
 15  adr                             119386 non-null  float64
 16  required_car_parking_spaces     119386 non-null  int64  
 17  total_of_special_requests       119386 non-null  int64  
dtypes: float64(2), int64(16)
memory usage: 17.3 MB
```

Logistic Regression

In [23]:
```
y = newhb['is_canceled']
x = newhb.drop(['is_canceled'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_
model = LogisticRegressionCV(Cs=10, cv=5, random_state=42)
result = model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print(model.score(X_test,y_test))
```

```
C:\Users\mtoke\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on
  n_iter_i = _check_optimize_result(
C:\Users\mtoke\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on
  n_iter_i = _check_optimize_result(
C:\Users\mtoke\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on
  n_iter_i = _check_optimize_result(
C:\Users\mtoke\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on
  n_iter_i = _check_optimize_result(
C:\Users\mtoke\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on
  n_iter_i = _check_optimize_result(
```

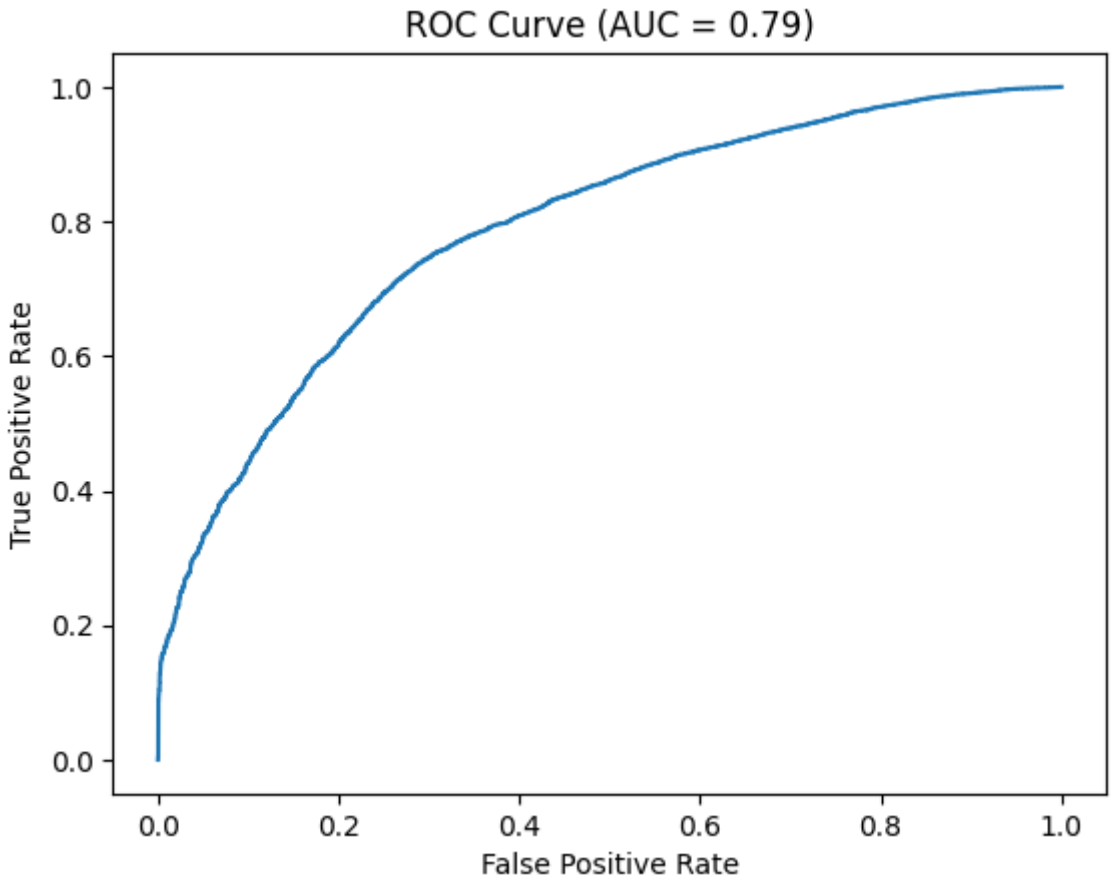
```
0.732473406482955
```

```
C:\Users\mtoke\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on
  n_iter_i = _check_optimize_result(
```

In [24]:
```
report = classification_report(y_test, y_pred)

print(report)
```

```
              precision    recall  f1-score   support

           0       0.74      0.89      0.81     14973
           1       0.71      0.48      0.57      8905

    accuracy                           0.73     23878
   macro avg       0.73      0.68      0.69     23878
weighted avg       0.73      0.73      0.72     23878
```

In [27]:
```
fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:, 1])

roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr)

roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])

fig, ax = plt.subplots()
roc_display.plot(ax=ax)

ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title(f'ROC Curve (AUC = {roc_auc:.2f})')
plt.show()
```



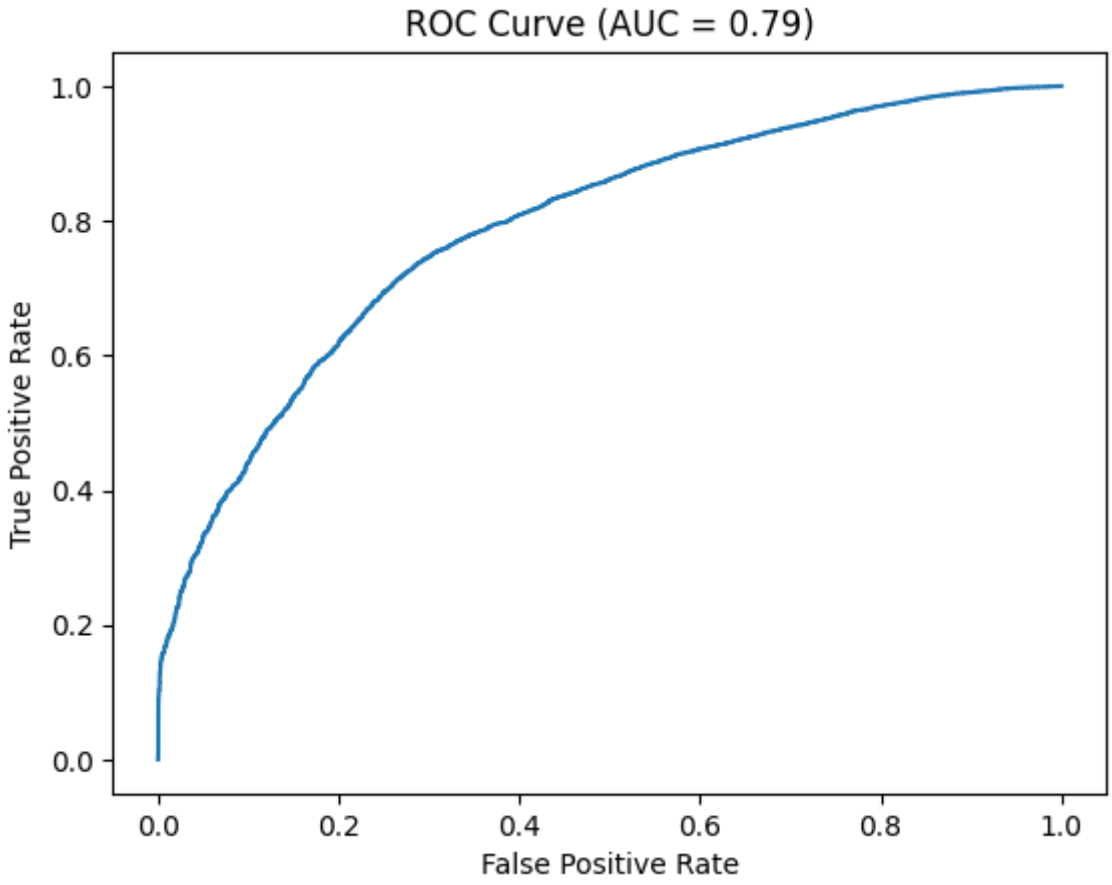
While Logistic Regression provides an average score of 0.73, we can see that our AUC value is 0.79, indicating that it successfully distinguishes between positive and negative outcomes.

Decision Tree

In [30]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

y = newhb['is_canceled']
x = newhb.drop(['is_canceled'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_
model1 = DecisionTreeClassifier()
result = model1.fit(X_train,y_train)
y_pred = model1.predict(X_test)
print(model1.score(X_test,y_test))

from sklearn.metrics import classification_report
rapor = classification_report(y_test, y_pred)
print(rapor)
fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:, 1])

roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr)

roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])

fig, ax = plt.subplots()
roc_display.plot(ax=ax)
```

```
ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title(f'ROC Curve (AUC = {roc_auc:.2f})')
plt.show()
```

```
0.8035430103023704
              precision    recall  f1-score   support

           0       0.85      0.84      0.84     14973
           1       0.73      0.75      0.74      8905

    accuracy                           0.80     23878
   macro avg       0.79      0.79      0.79     23878
weighted avg       0.80      0.80      0.80     23878
```



When the score of the Decision Tree model increases to 0.80, the AUC value remains the same. This indicates that the Decision Tree algorithm makes more accurate predictions compared to Logistic Regression.

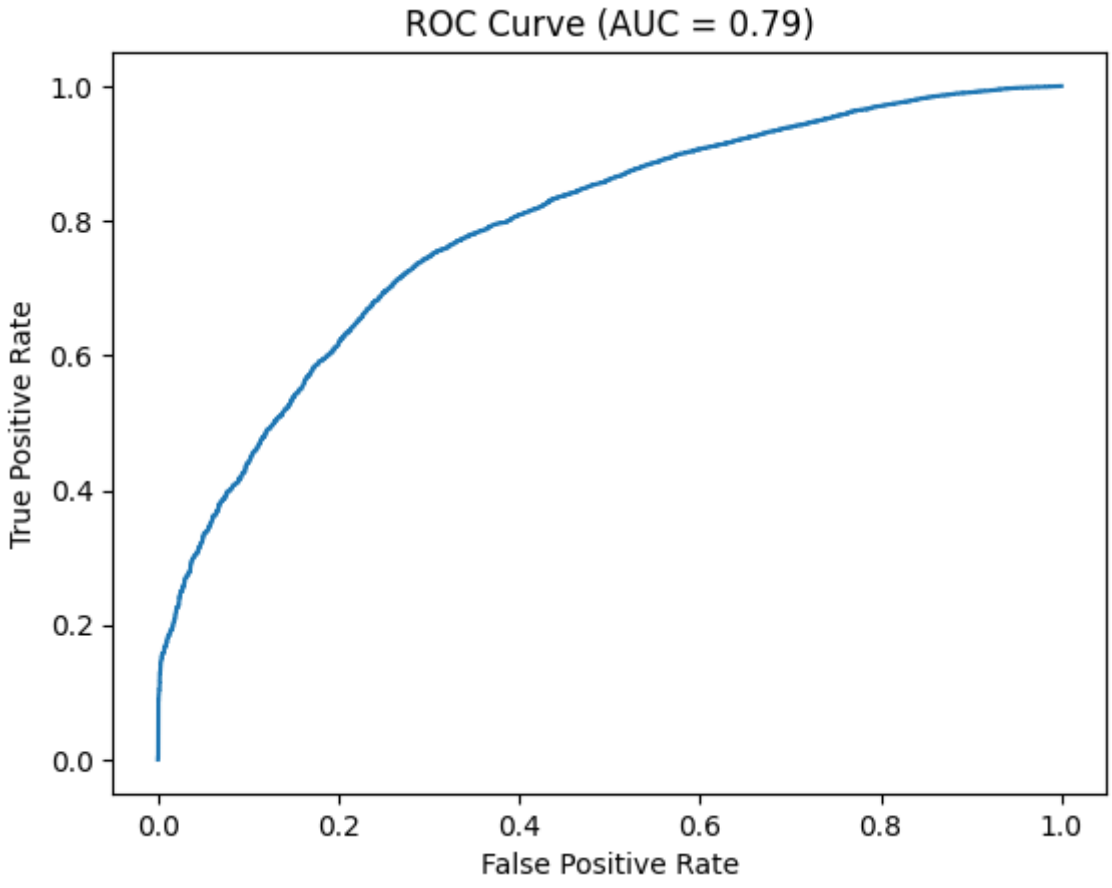
Random Forest

In [33]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
y = newhb['is_canceled']
x = newhb.drop(['is_canceled'], axis=1)
X_train2, X_test2, y_train2, y_test2 = train_test_split(x, y, test_size=0.2, rar
model2 = RandomForestClassifier()
```

```
result = model2.fit(X_train2,y_train2)
y_pred2 = model2.predict(X_test2)
print(model2.score(X_test2,y_test2))
rapor1 = classification_report(y_test2, y_pred2)
print(rapor1)
```

```
0.848940447273641
              precision    recall  f1-score   support

           0       0.85      0.92      0.88     14973
           1       0.85      0.72      0.78      8905

    accuracy                           0.85     23878
   macro avg       0.85      0.82      0.83     23878
weighted avg       0.85      0.85      0.85     23878
```

In [35]:
```
fpr, tpr, _ = roc_curve(y_test2, model.predict_proba(X_test2)[:, 1])

roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr)

roc_auc = roc_auc_score(y_test2, model.predict_proba(X_test2)[:, 1])

fig, ax = plt.subplots()
roc_display.plot(ax=ax)

ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title(f'ROC Curve (AUC = {roc_auc:.2f})')
plt.show()
```



As the score of the Random Forest model increases to 0.85, the AUC value remains the same. This indicates that the Random Forest algorithm provides more accurate predictions compared to the Decision Tree algorithm and Logistic Regression.