tour-and-travel-coustomer-pred

September 29, 2024

[70]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
```

[71]:
```
df = pd.read_csv('/content/Customertravel.csv')
df
```

[71]:

	Age	FrequentFlyer	AnnualIncomeClass	ServicesOpted	AccountSyncedToSocialMedia	BookedHotelOrNot	Target
0	34	No	Middle Income	6	No	Yes	0
1	34	Yes	Low Income	5	Yes	No	1
2	37	No	Middle Income	3	Yes	No	0
3	30	No	Middle Income	2	No	No	0
4	30	No	Low Income	1	No	No	0
..	…	…	…	…	…	…	…
949	31	Yes	Low Income	1	No	No	0
950	30	No	Middle Income	5	No	Yes	0
951	37	No	Middle Income	4	No	No	0
952	30	No	Low Income	1	Yes	Yes	0
953	31	Yes	High Income	1	No	No	0

[954 rows x 7 columns]

[72]:
```
df.head()
```

[72]:

	Age	FrequentFlyer	AnnualIncomeClass	ServicesOpted	AccountSyncedToSocialMedia	BookedHotelOrNot	Target
0	34	No	Middle Income	6	No	Yes	0
1	34	Yes	Low Income	5	Yes	No	1
2	37	No	Middle Income	3	Yes	No	0
3	30	No	Middle Income	2	No	No	0
4	30	No	Low Income	1	No	No	0

[73]:
```
df.tail()
```

[73]:

	Age	FrequentFlyer	AnnualIncomeClass	ServicesOpted	AccountSyncedToSocialMedia	BookedHotelOrNot	Target
949	31	Yes	Low Income	1	No	No	0
950	30	No	Middle Income	5	No	Yes	0
951	37	No	Middle Income	4	No	No	0
952	30	No	Low Income	1	Yes	Yes	0
953	31	Yes	High Income	1	No	No	0

[74]:
```
df.shape
```

[74]:
```
(954, 7)
```

[75]:
```
df.describe()
```

[75]:

	Age	ServicesOpted	Target
count	954.000000	954.000000	954.000000
mean	32.109015	2.437107	0.234801
std	3.337388	1.606233	0.424097
min	27.000000	1.000000	0.000000
25%	30.000000	1.000000	0.000000
50%	31.000000	2.000000	0.000000
75%	35.000000	4.000000	0.000000
max	38.000000	6.000000	1.000000

[76]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 954 entries, 0 to 953
Data columns (total 7 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   Age                         954 non-null    int64 
 1   FrequentFlyer               954 non-null    object
 2   AnnualIncomeClass           954 non-null    object
 3   ServicesOpted               954 non-null    int64 
 4   AccountSyncedToSocialMedia  954 non-null    object
 5   BookedHotelOrNot            954 non-null    object
 6   Target                      954 non-null    int64 
dtypes: int64(3), object(4)
memory usage: 52.3+ KB
```

[77]:
```
df.isna().sum()
```

[77]:
```
Age                           0
FrequentFlyer                 0
AnnualIncomeClass             0
ServicesOpted                 0
AccountSyncedToSocialMedia    0
BookedHotelOrNot              0
Target                        0
dtype: int64
```

One Hot Encoding

[78]:
```
df2 = pd.
↪get_dummies(df,columns=['FrequentFlyer','AccountSyncedToSocialMedia','BookedHotelOrNot'],dr
```

[79]:
```
label_encoder = LabelEncoder()
df2['AnnualIncomeClass'] = label_encoder.fit_transform(df2['AnnualIncomeClass'])
```

[80]:
```
df2
```

[80]:

	Age	AnnualIncomeClass	ServicesOpted	Target	FrequentFlyer_No Record	\
0	34	2	6	0	False	
1	34	1	5	1	False	
2	37	2	3	0	False	
3	30	2	2	0	False	
4	30	1	1	0	False	
..	…	…	…	…	…	
949	31	1	1	0	False	
950	30	2	5	0	False	

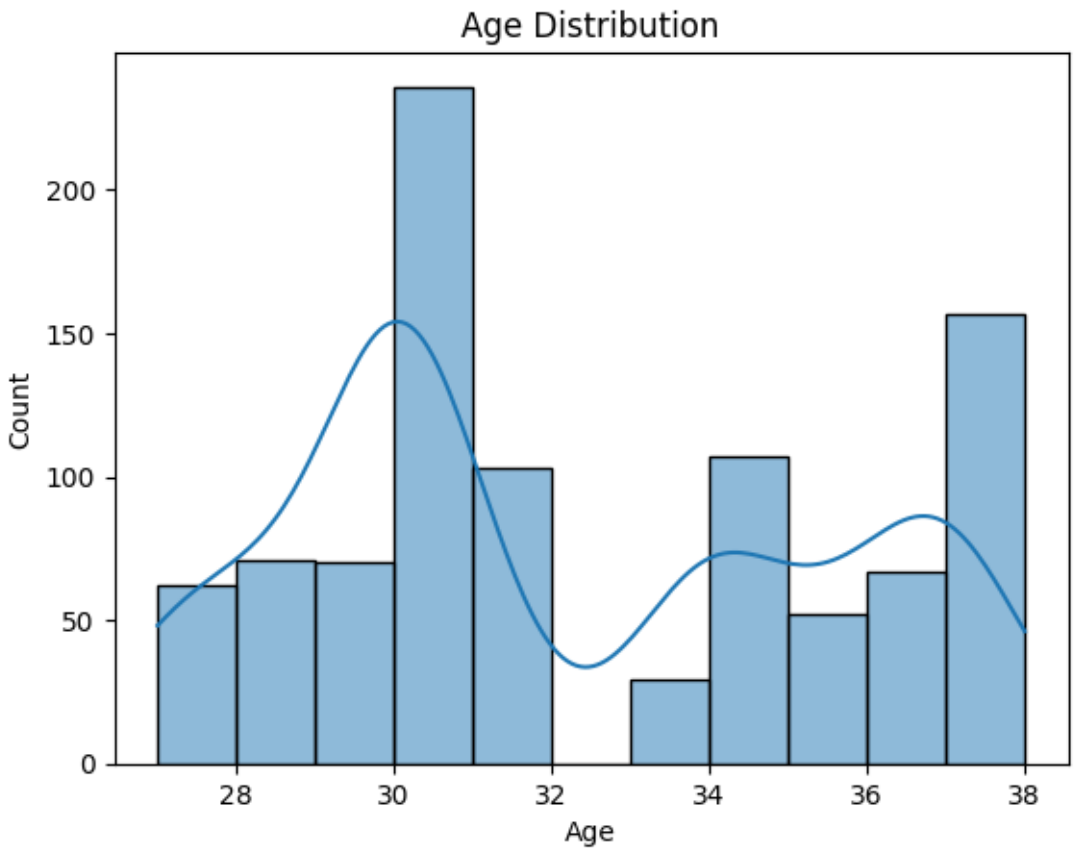
```
951   37                  2               4       0                    False
952   30                  1               1       0                    False
953   31                  0               1       0                    False

     FrequentFlyer_Yes  AccountSyncedToSocialMedia_Yes  BookedHotelOrNot_Yes
0                False                           False                 True
1                 True                            True                False
2                False                            True                False
3                False                           False                False
4                False                           False                False
..                 …                               …                    …
949               True                           False                False
950              False                           False                 True
951              False                           False                False
952              False                            True                 True
953               True                           False                False

[954 rows x 8 columns]
```

[81]:
```
df['Age'].value_counts()
```

[81]:
```
Age
30    236
37    126
34    107
31    103
28     71
29     70
36     67
27     62
35     52
38     31
33     29
Name: count, dtype: int64
```

Age Distribution

[82]:
```
sns.histplot(df['Age'], kde=True)
plt.title('Age Distribution')
plt.show()
```



[83]:
```
df['Target'].value_counts()
```

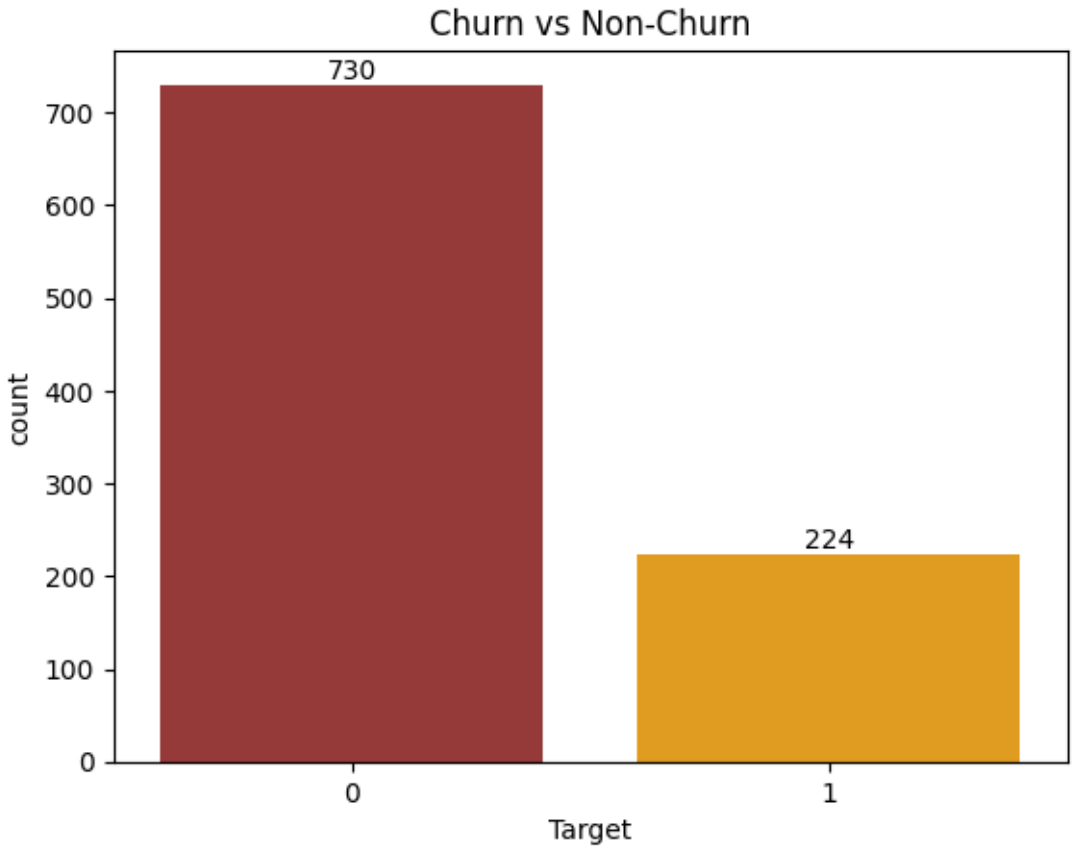
[83]:
```
Target
0    730
1    224
Name: count, dtype: int64
```

Taget Distribution (Churn VS Non-Churn)

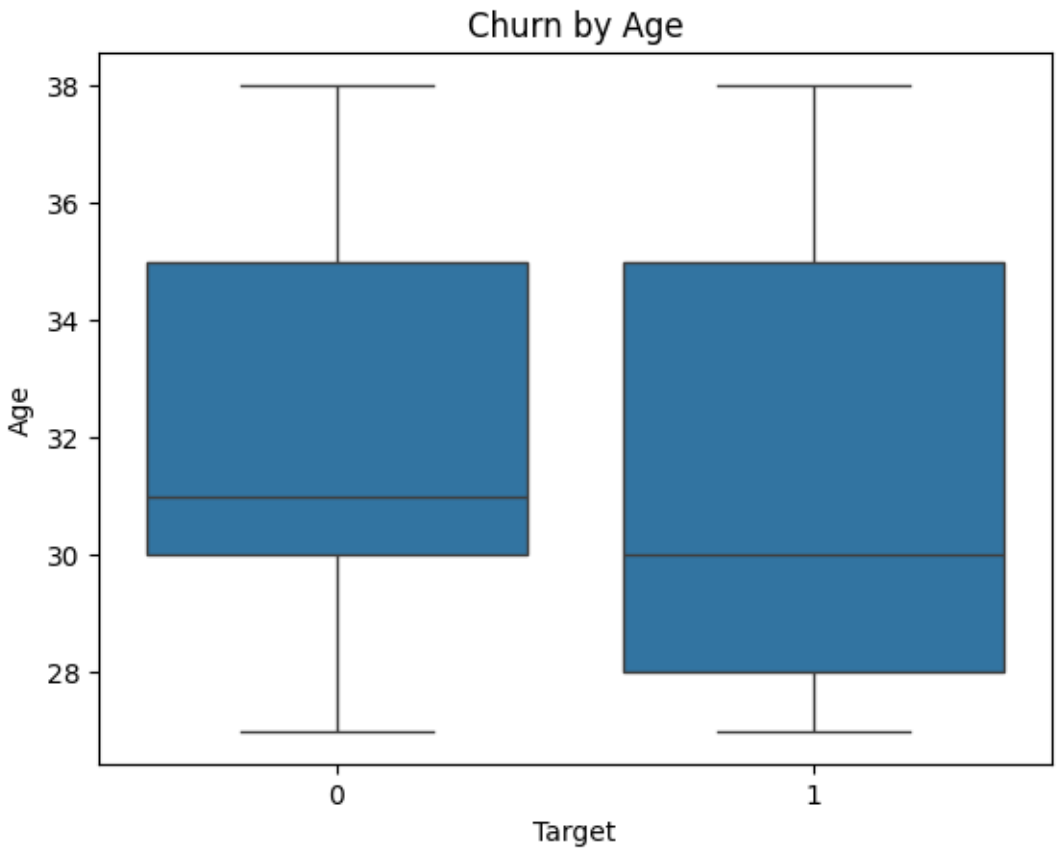
[84]:
```
ax=sns.countplot(x='Target', data=df2,palette=['brown','orange'])
plt.title('Churn vs Non-Churn')
for x in ax.containers:
  ax.bar_label(x,rotation=0)
plt.show()
```

```
<ipython-input-84-6f34037d67a4>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```

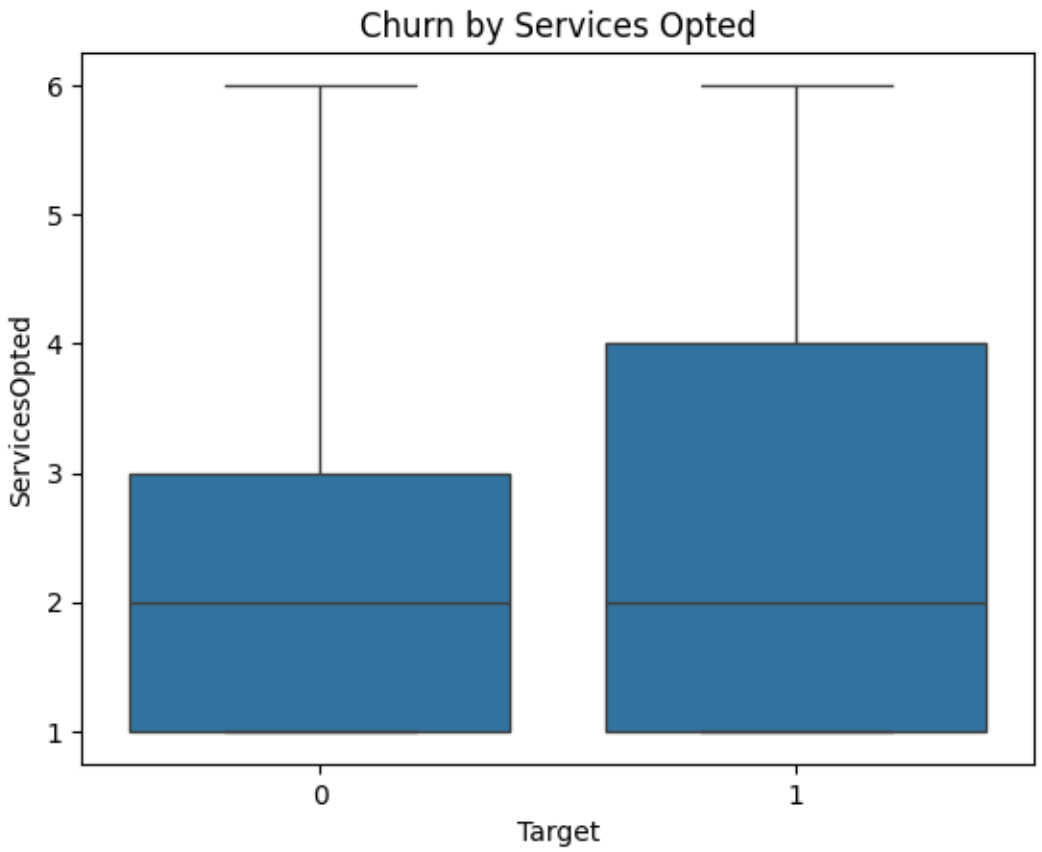
```
ax=sns.countplot(x='Target', data=df2,palette=['brown','orange'])
```



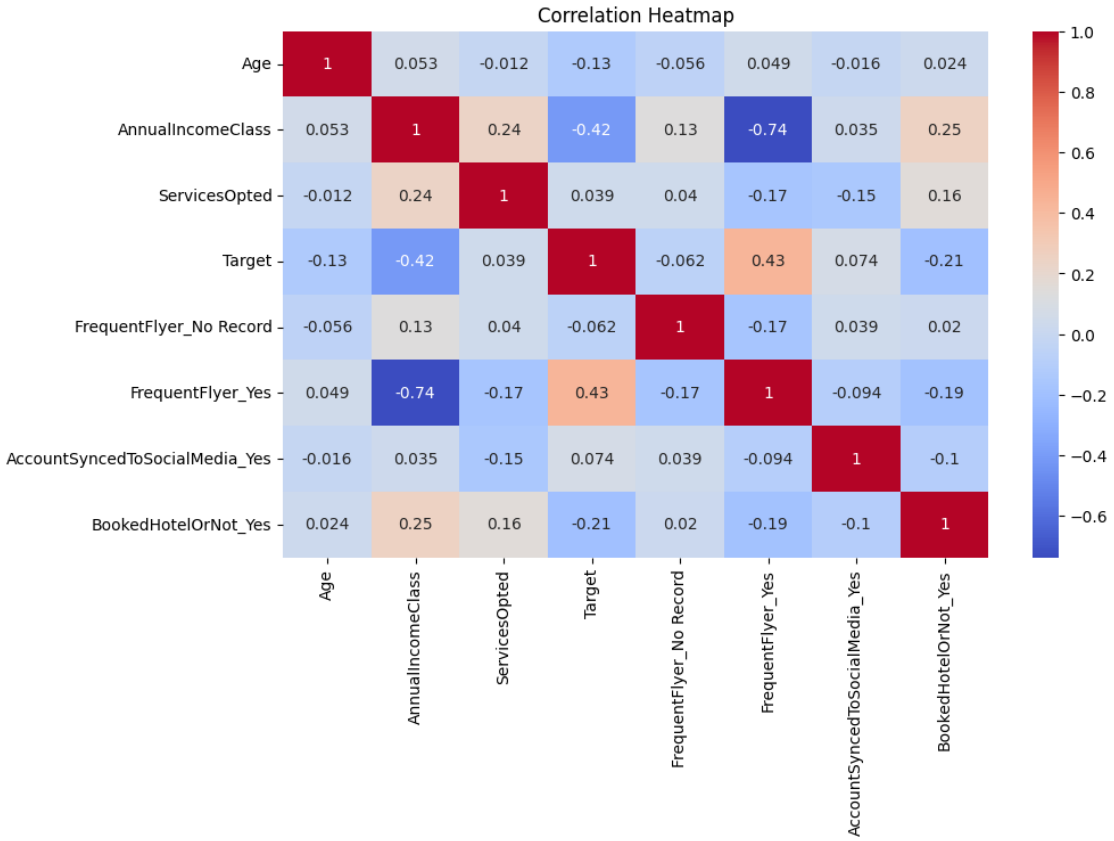
[85]:
```
# Age vs Churn
sns.boxplot(x='Target', y='Age', data=df)
plt.title('Churn by Age')
plt.show()
```



[86]:
```
# ServicesOpted vs Churn
sns.boxplot(x='Target', y='ServicesOpted', data=df)
plt.title('Churn by Services Opted')
plt.show()
```



[87]:
```
# Correlation matrix
plt.figure(figsize=(10,6))
sns.heatmap(df2.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



[88]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

[89]:
```
X = df2.drop(columns=['Target'])
y = df2['Target']

# Fit the Random Forest model
rf_model = RandomForestClassifier()
rf_model.fit(X, y)

# Get feature importance
importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

# Display feature importance
print(importance)
```

```
                       Feature  Importance
0                          Age    0.298319
```

```
2                   ServicesOpted    0.244906
4               FrequentFlyer_Yes    0.158610
1               AnnualIncomeClass    0.152560
5  AccountSyncedToSocialMedia_Yes    0.089868
6            BookedHotelOrNot_Yes    0.045187
3         FrequentFlyer_No Record    0.010549
```

[90]:
```
selected_features = ['Age', 'ServicesOpted', 'AnnualIncomeClass',␣
 ↪'FrequentFlyer_Yes',
                     'AccountSyncedToSocialMedia_Yes', 'BookedHotelOrNot_Yes']

X_selected = df2[selected_features]
y = df2['Target']

# Perform train/test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.
 ↪3, random_state=42)
```

[91]:
```
from xgboost import XGBClassifier

# Initialize and train XGBoost
xgb = XGBClassifier(scale_pos_weight=len(y_train[y_train == 0]) /␣
 ↪len(y_train[y_train == 1]))  # Handle imbalance
xgb.fit(X_train, y_train)

# Predict and evaluate
y_pred_xgb = xgb.predict(X_test)
print(f"XGBoost Accuracy: {accuracy_score(y_test, y_pred_xgb)}")
print(classification_report(y_test, y_pred_xgb))
```

```
XGBoost Accuracy: 0.8954703832752613
              precision    recall  f1-score   support

           0       0.96      0.90      0.93       219
           1       0.74      0.87      0.80        68

    accuracy                           0.90       287
   macro avg       0.85      0.89      0.86       287
weighted avg       0.90      0.90      0.90       287
```