تمرین کامپیوتری سوم هوش مصنوعی

امیرمرتضی رضائی – 810003004

آماده‌سازی محیط و تحلیل اکتشافی داده:

مجموعه داده‌ها به‌صورت زیر به فرمت DataFrame در Pandas تبدیل شد. در ادامه نیز ساختار کلی داده‌ها با استفاده از متدهای info و describe بدست آمد:

```
df = pd.read_csv('hotels_in_europe.csv')
df.info()
df.describe()
```

✓ 0.0s

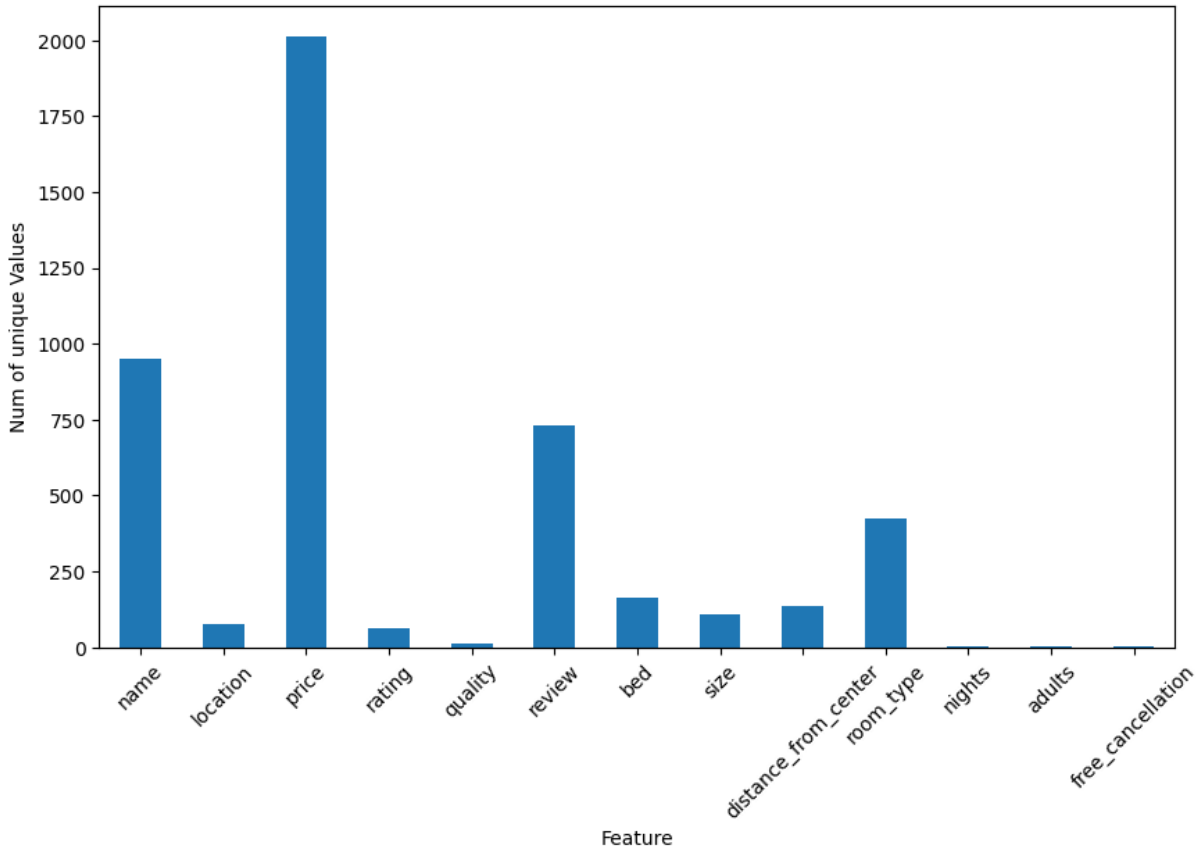
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7322 entries, 0 to 7321
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   name                  7322 non-null   object 
 1   location              7322 non-null   object 
 2   price                 7322 non-null   object 
 3   rating                7129 non-null   float64
 4   quality               7169 non-null   object 
 5   review                7169 non-null   object 
 6   bed                   7299 non-null   object 
 7   size                  2454 non-null   object 
 8   distance_from_center  7322 non-null   float64
 9   room_type             7322 non-null   object 
 10  nights                7322 non-null   object 
 11  adults                7322 non-null   object 
 12  free_cancellation     583 non-null    object 
dtypes: float64(2), object(11)
memory usage: 743.8+ KB
```

	rating	distance_from_center
count	7129.000000	7322.000000
mean	7.742783	24.699399
std	1.128965	82.440210
min	1.000000	0.500000
25%	7.300000	2.200000
50%	8.000000	3.700000
75%	8.400000	5.800000
max	10.000000	500.000000

در ادامه، نمودار تعداد مقادیر منحصر به‌فرد به ازای هر یک از ویژگی‌ها رسم شد:

```
#plotting the num of unique vals of each feature:
unique_values = df.nunique()
unique_values.plot(kind='bar', figsize=(10, 6))
plt.xlabel('Feature')
plt.ylabel('Num of unique Values')
plt.xticks(rotation=45)
plt.show()
```

✓ 0.1s



همان‌طور که مشاهده می‌گردد، کمترین تعداد مقادیر منحصر به‌فرد متعلق به کنسلی رایگان، تعداد بزرگسالان، تعداد شب‌های اقامت و کیفیت و rating و لوکشین می‌باشد.

نمودارهای وابستگی و ارتباطات میان ویژگی‌ها را پس از پیش‌پردازش داده‌ها رسم می‌کنیم. چون اکنون به مقادیر عددی تمام ویژگی‌ها دسترسی نداریم.

پیش‌پردازش دادگان و مهندسی ویژگی‌ها:

در ابتدا، تعدادی ستون را به دیتافریم اضافه می‌کنیم:

1- ستون city که شهر هر هتل را از لوکیشن آن استخراج کرده‌ایم.

2- ستون Numetic_Price که تنها مقدار عددی هزینه هر هتل را بدون هیچ پسوند و علامتی در خود دارد.

3- ستون adults_count که تنها تعداد بزرگسالان را بدون هیچ پسوند و علامتی در خود دارد.

4- ستون nights_count که تنها تعداد شب‌ها را بدون هیچ پسوند و علامتی در خود دارد.

5- ستون reviews_count که تنها تعداد نظرات را بدون هیچ پسوندی در خود دارد.

6- در ستون distance، به ترتیب مقادیر 1و 2و 3و 4 را به داده هایی با فاصله از مرکز کم تر از چارک اول، بین چارک اول و دوم، بین چارک دوم و سوم و بیش‌تر از چارک سوم اختصاص می‌دهیم.

```
def assign_value(x):
    if x <= distance_Q1:
        return 1
    elif distance_Q1 < x <= distance_Q2:
        return 2
    elif distance_Q2 < x <= distance_Q3:
        return 3
    else:
        return 4
```

0.0s

```
df['city'] = df['location'].str.split(',').str[-1].str.strip()

df['Numeric_Price'] = df['price'].str.replace(r'[^\d,]', '', regex=True)
df['Numeric_Price'] = df['Numeric_Price'].str.replace(',', '').astype(float)

df['adults_count'] = df['adults'].str.extract(r'(\d+)').astype(int)

df['nights_count'] = df['nights'].str.extract(r'(\d+)').astype(int)

df['reviews_count'] = df['review'].str.replace(r'[^\d]', '', regex=True)

df['distance'] = df['distance_from_center'].apply(assign_value)
```

حال، تعدادی از ستون‌ها را حذف می‌کنیم:

1- ستون name: چون تعداد مقادیر منحصر به‌فرد به‌نسبت بالایی دارد و احتمالا کمکی زیادی در پیش‌بینی قیمت نخواهد کرد.
2- ستون quality: چون نمودی از آن در ستون rating وجود دارد و این دو ستون شباهت بالایی به‌هم دارند.
3- ستون size : همانطور که در بخش قبل دیدیم، حدود دو سوم مقادیر این ستون خالی هستند و مقداری ندارند.
4- ستون bed : چون نمودی از آن در ستون adults وجود دارد و این دو ستون شباهت بالایی به‌هم دارند.

```
#deleting some columns
df.drop(columns=['name'], inplace=True)
df.drop(columns=['quality'], inplace=True)
df.drop(columns=['size'], inplace=True)
df.drop(columns=['bed'], inplace=True)
```

در ادامه، داده‌های دسته‌بندی شده را به مقادیر عددی تبدیل می‌کنیم:

1- به هر شهر در ستون city، مقداری را اختصاص داده و در ستون city_numeric ذخیره می‌کنیم.
2- به هر تایپ در ستون room_type، مقداری را اختصاص داده و در ستون room_type_numeric ذخیره می‌کنیم.
3- برای رزرواسیون‌هایی که قابلیت کنسلی رایگان دارند، مقدار 1 و برای باقی مقدار 0 را اختصاص داده و این مقادیر را در ستون free_cancellation_numeric ذخیره می‌کنیم.

```
#assigning values to non-numeric datas
label_encoder = LabelEncoder()
df['city_numeric'] = label_encoder.fit_transform(df['city'])

df['room_type_numeric'] = label_encoder.fit_transform(df['room_type'])

cancelation_map = {'free_cancellation': 1}
df['free_cancellation_numeric'] = df['free_cancellation'].map(cancelation_map).fillna(0)
```

حالا داده‌های از دست رفته را تکمیل می‌کنیم.

1- مقادیر از دست‌رفته ستون reviews_count را با مقدار ثابت صفر تکمیل می‌کنیم.

2- مقادیر از دست‌رفته ستون rating را با مقدار میانه‌ی این ستون تکمیل می‌کنیم.

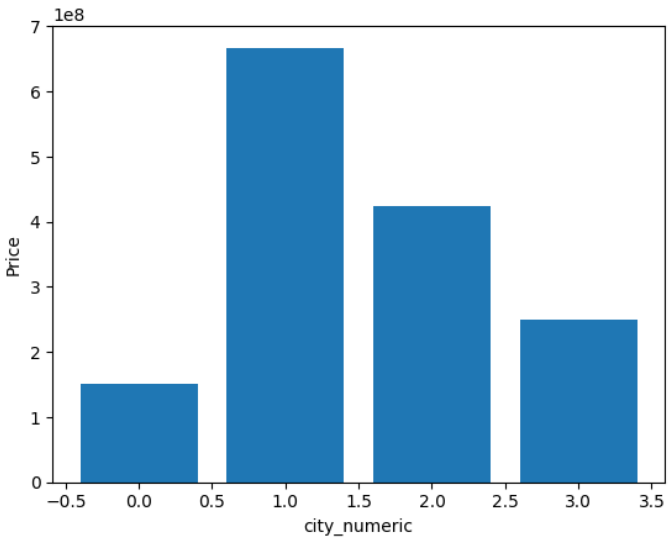
```
df['reviews_count'] = df['reviews_count'].fillna(0)
df['reviews_count'] = df['reviews_count'].astype(int)

df['rating'] = df['rating'].fillna(df['rating'].median())
```

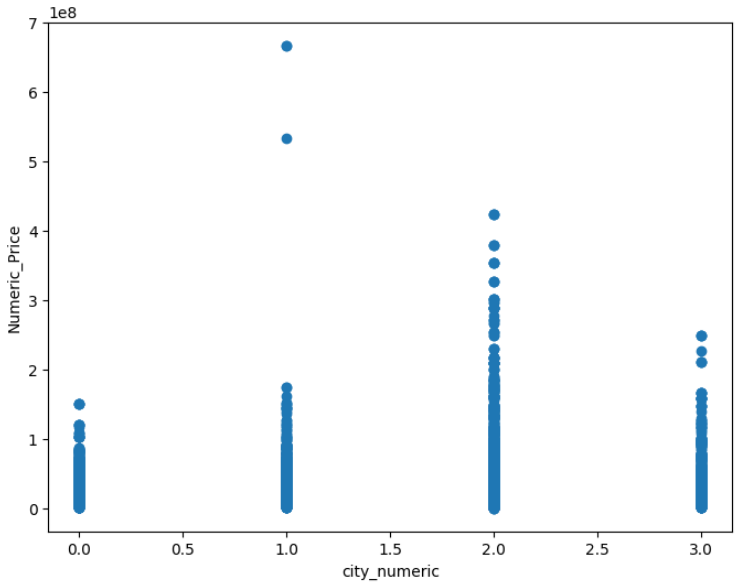
اکنون دو برچسب 0 و 1 را با توجه به حدود قیمتی هتل‌ها، به آن‌ها اختصاص می‌دهیم. اگر قیمت هتلی از میانه بیشتر بود، برچسب 1 و در غیر این‌صورت برچسب 0 را می‌گیرد.

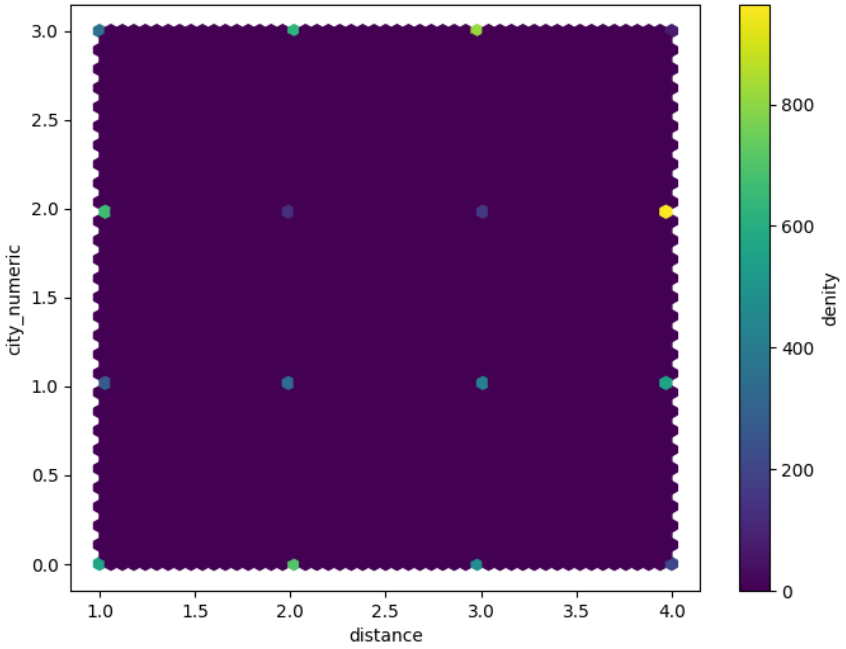
```
def assign_label(x):
    the_median = df['Numeric_Price'].median()
    if x >= the_median:
        return 1
    else:
        return 0

df['price_Label'] = df['Numeric_Price'].apply(assign_label)
```

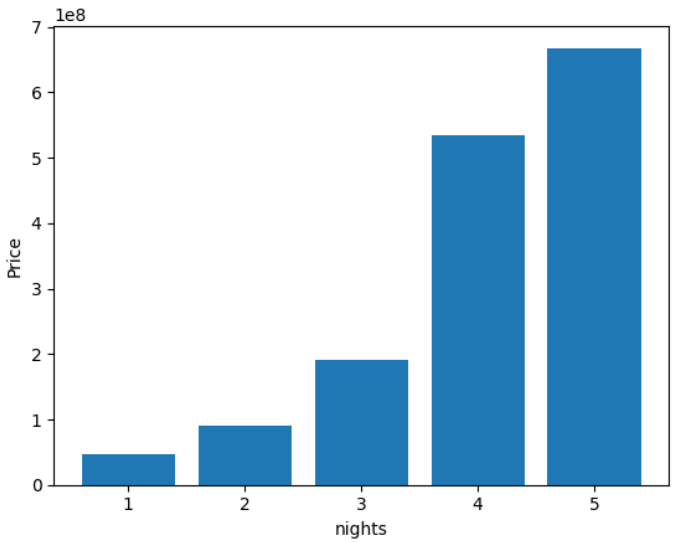
رسم نمودار های ارتباطات میان ویژگی‌ها:

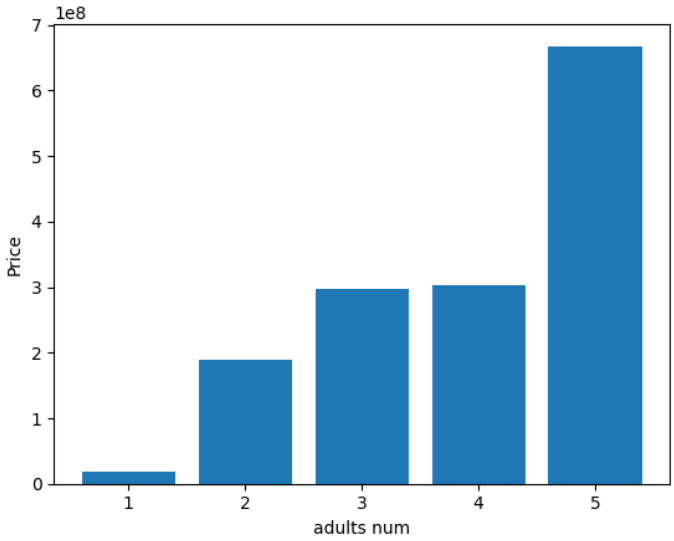
تعدادی از نمودارهای رسم شده به شرح زیر است:

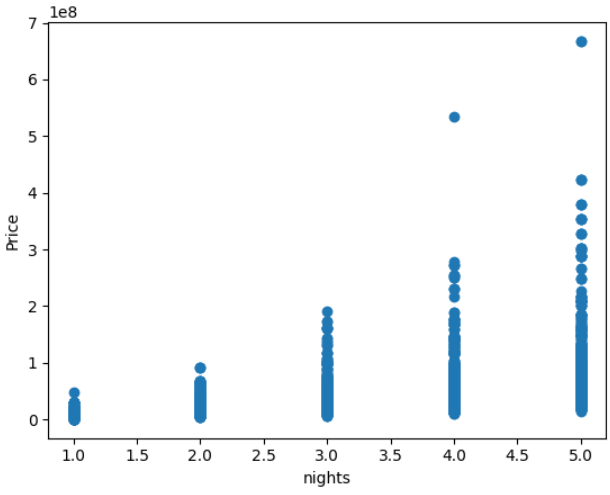


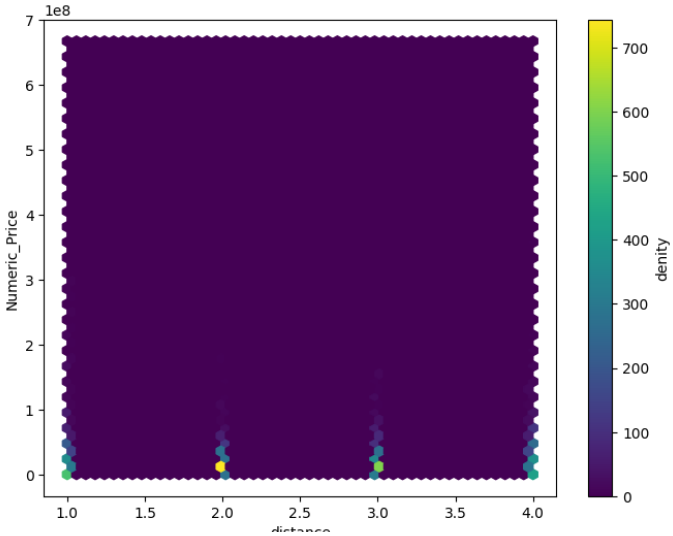


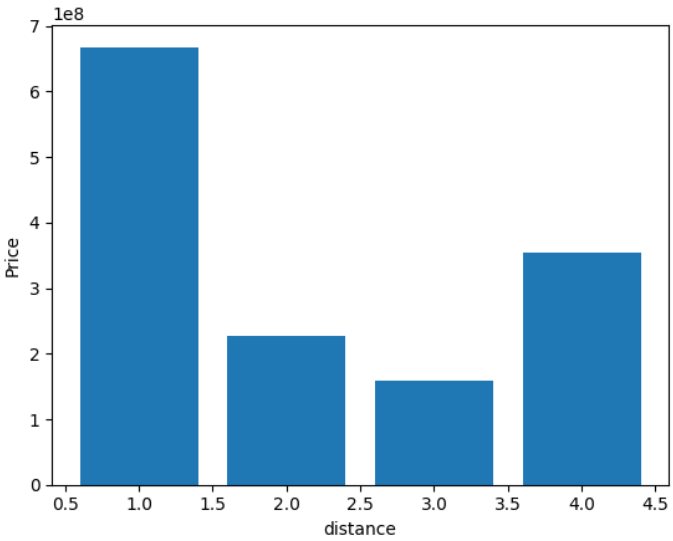

city_numeric
distance
denity
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
800
600
400
200
0

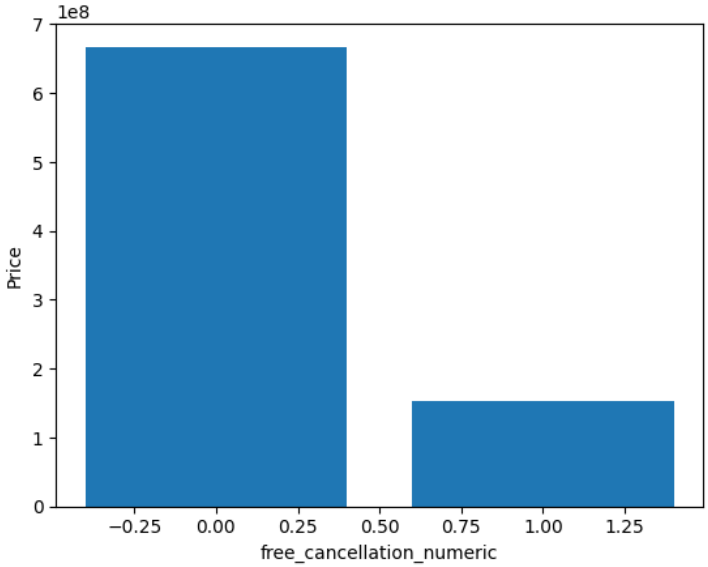

1e8
Price
nights
7
6
5
4
3
2
1
0
1
2
3
4
5

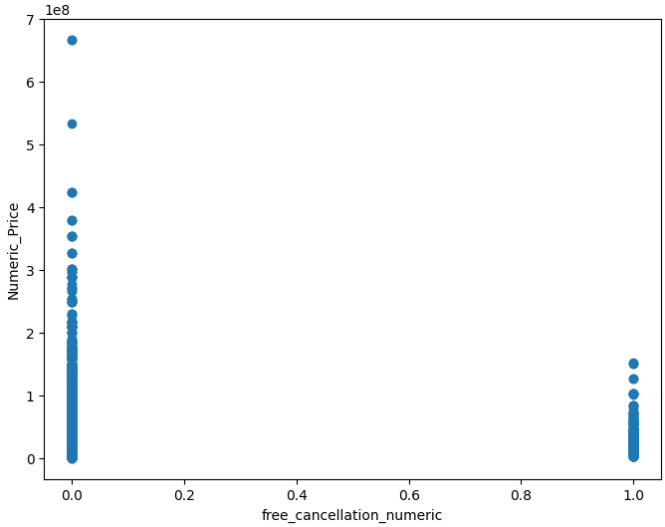

1e8
Price
adults num
7
6
5
4
3
2
1
0
1
2
3
4
5

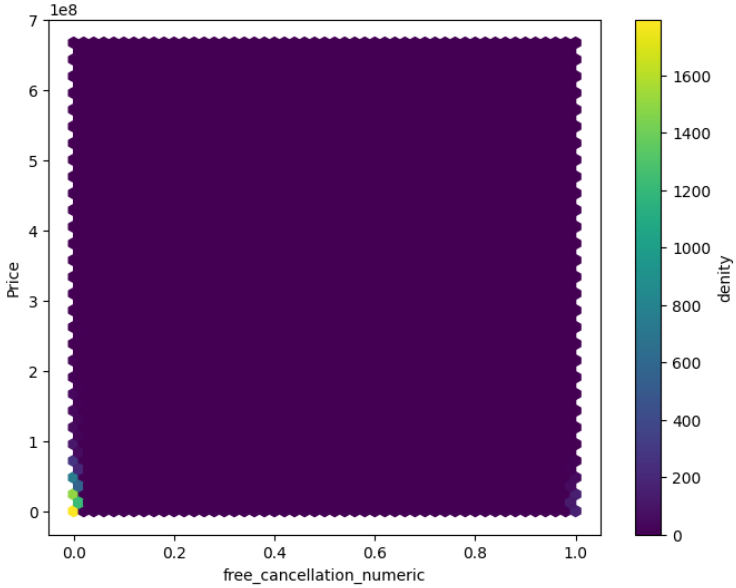

1e8
Price
nights
7
6
5
4
3
2
1
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


1e8
Numeric_Price
distance
denity


1e8
Price
distance


1e8
Price
free_cancellation_numeric


1e8
Numeric_Price
free_cancellation_numeric


1e8
Price
free_cancellation_numeric
denity

:Train-Test split

انجام شد.

:Normalization/Standardization

با استفاده از متد MinMaxScaler انجام شد:

```
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)
# Convert back to DataFrame
X_train_normalized = pd.DataFrame(X_train_normalized, columns=X_train.columns)
X_test_normalized = pd.DataFrame(X_test_normalized, columns=X_test.columns)
```

متد های ارزیابی:

- ✓ ماتریس درهم ریختگی -> TP , TN , FP , FN
- ✓ ReCall:

$$Recall = \frac{TP}{TP + FN}$$

- ✓ Precision:

$$Precision = \frac{TP}{TP + FP}$$

- ✓ F1-Score:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- ✓ Accuracy:

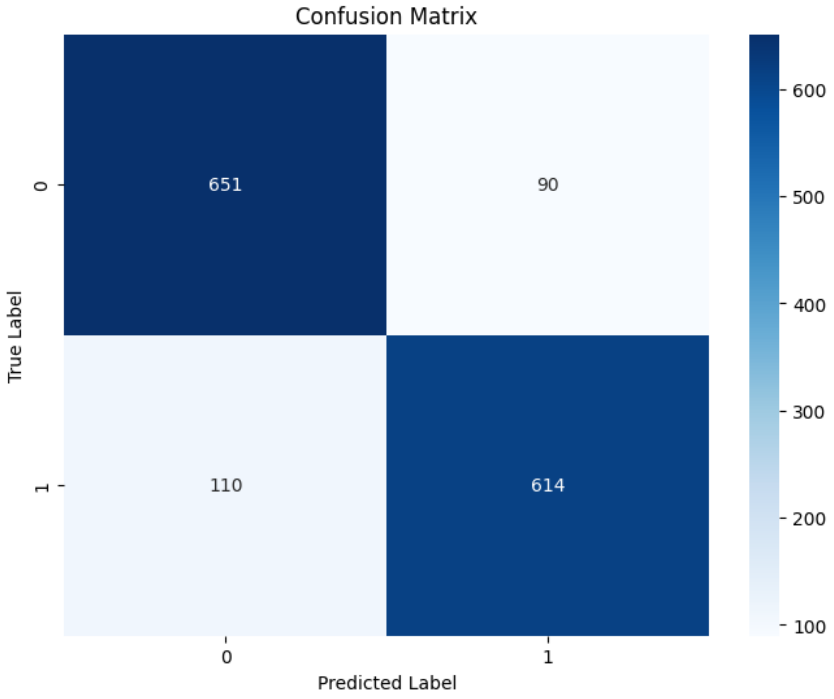
$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

:Gaussian Naive Bayes

به‌صوت زیر انجام شد:

```
modelNB = GaussianNB()
modelNB.fit(X_train_normalized, y_train)
y_pred = modelNB.predict(X_test_normalized)
```

ارزیابی:

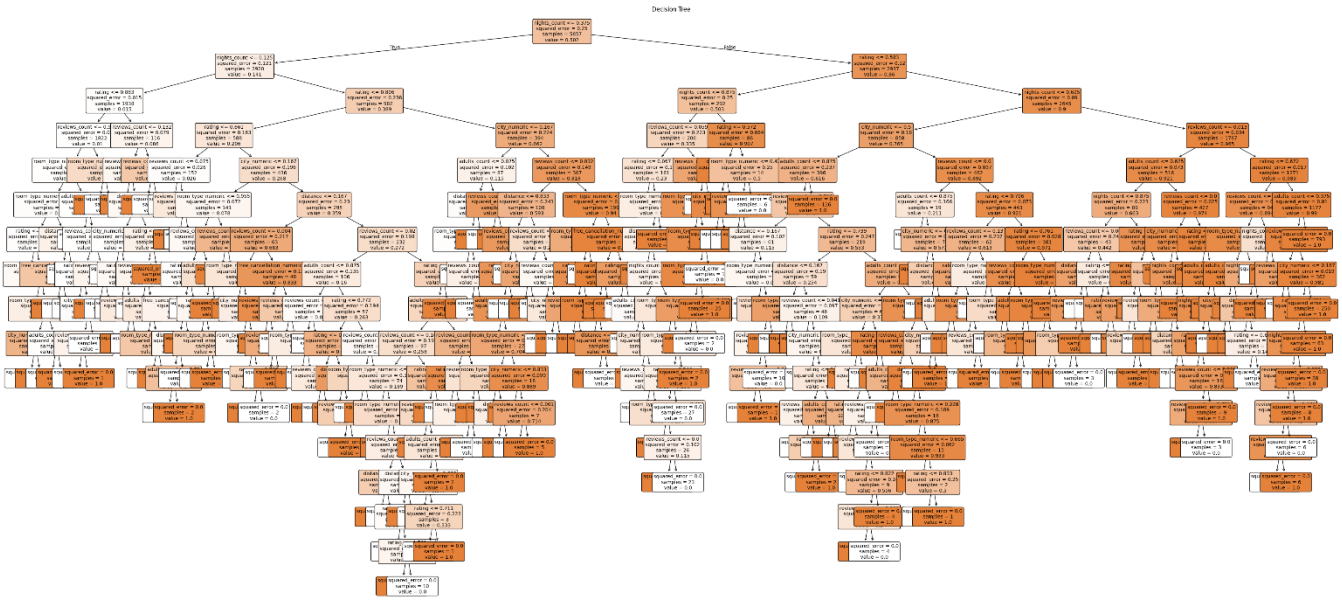



```
Recall: 84.8066%

Precision: 87.2159%

F1-Score: 0.8599

Accuracy: 86.35%

Precision (micro): 0.8635
Precision (macro): 0.8638
Precision (weighted): 0.8637

Recall (micro): 0.8635
Recall (macro): 0.8633
Recall (weighted): 0.8635

F1-Score (micro): 0.8635
F1-Score (macro): 0.8634
F1-Score (weighted): 0.8634
```

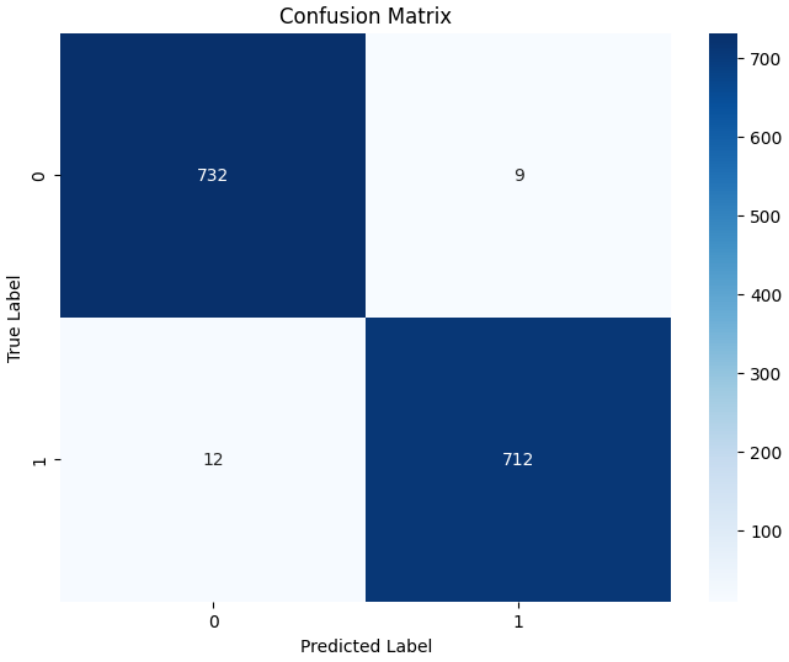
TN = 651 , TP = 614 , FN = 110 , FP = 90

:Decision Tree

به‌صوت زیر انجام شد:



ارزیابی:

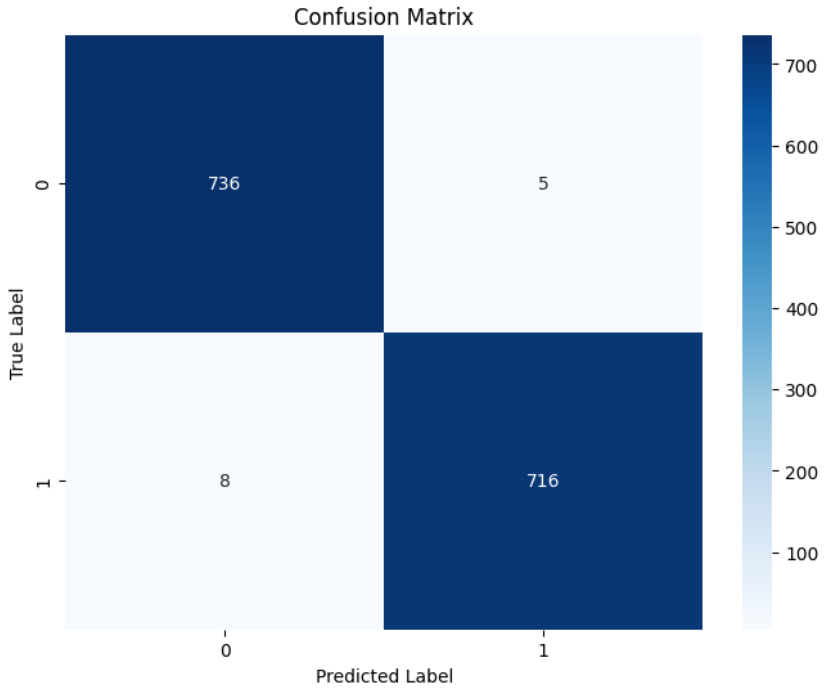



```
Recall: 98.3425%

Precision: 98.7517%

F1-Score: 0.9855

Accuracy: 98.57%

Precision (micro): 0.9857
Precision (macro): 0.9857
Precision (weighted): 0.9857

Recall (micro): 0.9857
Recall (macro): 0.9856
Recall (weighted): 0.9857

F1-Score (micro): 0.9857
F1-Score (macro): 0.9857
F1-Score (weighted): 0.9857
```

TN = 732 , TP = 712 , FN = 12 , FP = 9

:Random Forest

ارزیابی:

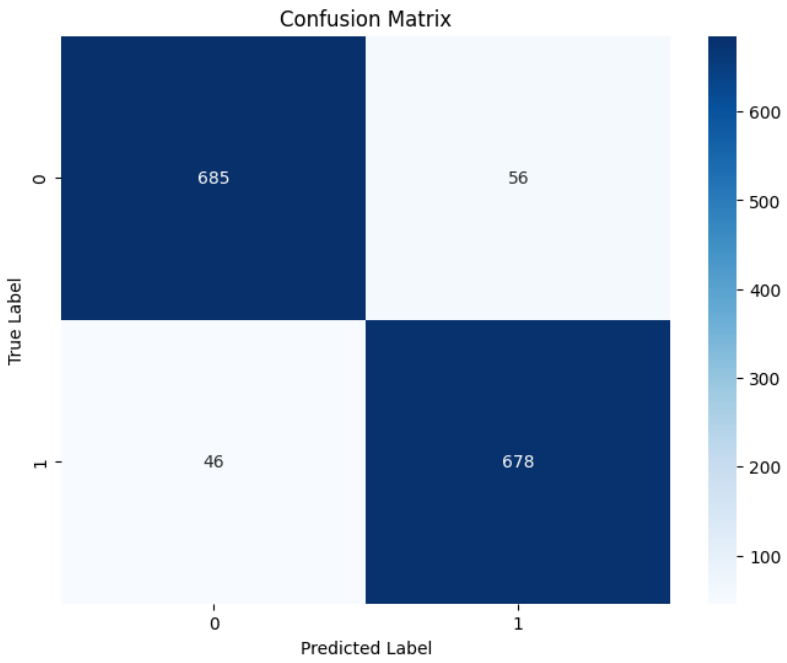



```
Recall: 98.8950%

Precision: 99.3065%

F1-Score: 0.9910

Accuracy: 99.11%

Precision (micro): 0.9911
Precision (macro): 0.9912
Precision (weighted): 0.9911

Recall (micro): 0.9911
Recall (macro): 0.9911
Recall (weighted): 0.9911

F1-Score (micro): 0.9911
F1-Score (macro): 0.9911
F1-Score (weighted): 0.9911
```

TN = 736 , TP = 716 , FN = 8 , FP = 5

:Adaptive Boosting

ارزیابی:

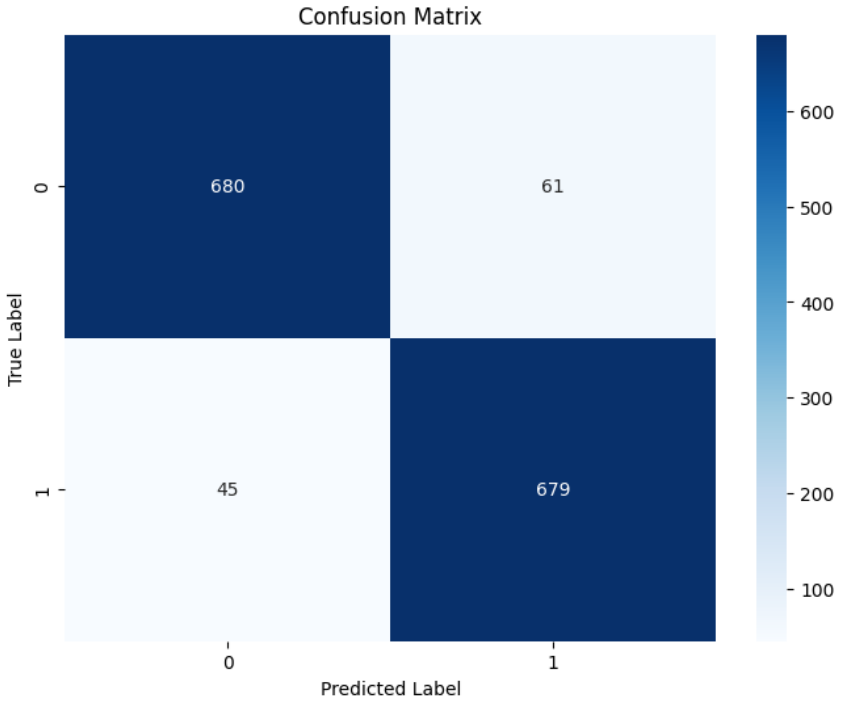



```
Recall: 93.6464%

Precision: 92.3706%

F1-Score: 0.9300

Accuracy: 93.04%

Precision (micro): 0.9304
Precision (macro): 0.9304
Precision (weighted): 0.9305

Recall (micro): 0.9304
Recall (macro): 0.9304
Recall (weighted): 0.9304

F1-Score (micro): 0.9304
F1-Score (macro): 0.9304
F1-Score (weighted): 0.9304
```

TN = 685 , TP = 678 , FN = 46 , FP = 56

:Adaptive Boosting, n_estimator = 50

ارزیابی:

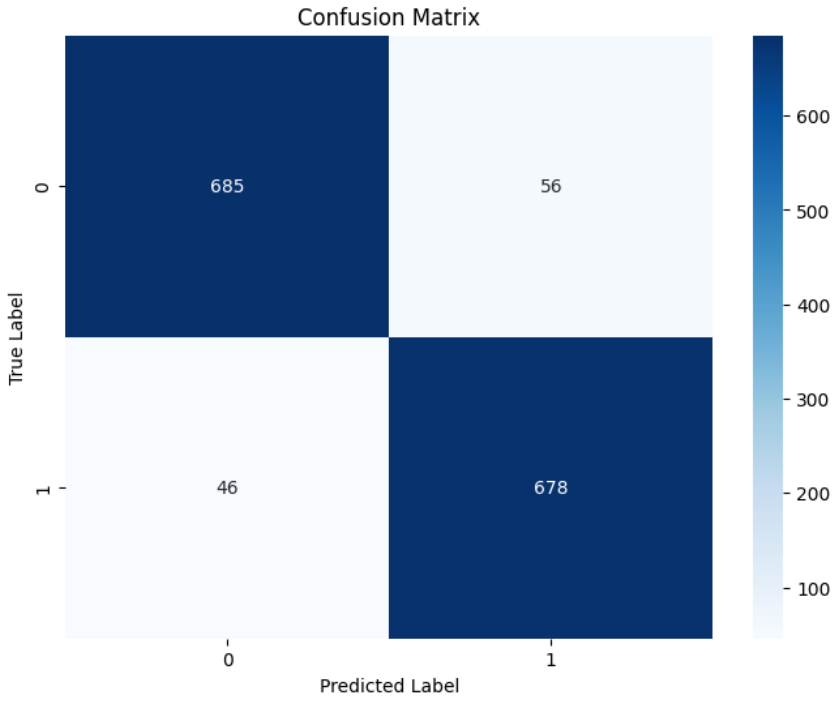



```
Recall: 93.7845%

Precision: 91.7568%

F1-Score: 0.9276

Accuracy: 92.76%

Precision (micro): 0.9276
Precision (macro): 0.9277
Precision (weighted): 0.9279

Recall (micro): 0.9276
Recall (macro): 0.9278
Recall (weighted): 0.9276

F1-Score (micro): 0.9276
F1-Score (macro): 0.9276
F1-Score (weighted): 0.9276
```

TN = 680 , TP = 679 , FN = 45 , FP = 61

:Adaptive Boosting, n_estimator = 100

ارزیابی:

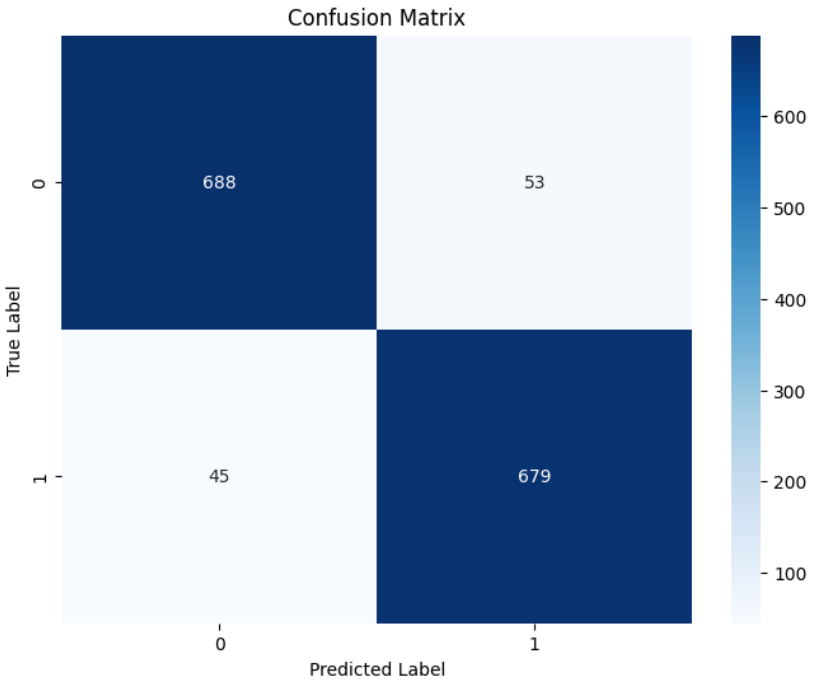



```
Recall: 93.6464%

Precision: 92.3706%

F1-Score: 0.9300

Accuracy: 93.04%

Precision (micro): 0.9304
Precision (macro): 0.9304
Precision (weighted): 0.9305

Recall (micro): 0.9304
Recall (macro): 0.9304
Recall (weighted): 0.9304

F1-Score (micro): 0.9304
F1-Score (macro): 0.9304
F1-Score (weighted): 0.9304
```

TN = 685 , TP = 678 , FN = 46 , FP = 56

:Adaptive Boosting, n_estimator = 200

ارزیابی:



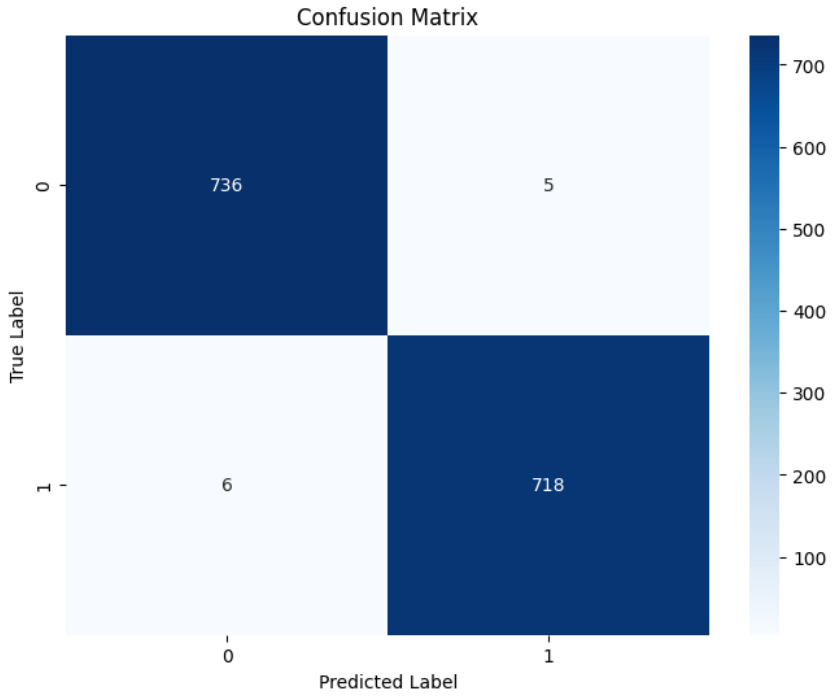

```
Recall: 93.7845%

Precision: 92.7596%

F1-Score: 0.9327

Accuracy: 93.31%

Precision (micro): 0.9331
Precision (macro): 0.9331
Precision (weighted): 0.9332

Recall (micro): 0.9331
Recall (macro): 0.9332
Recall (weighted): 0.9331

F1-Score (micro): 0.9331
F1-Score (macro): 0.9331
F1-Score (weighted): 0.9331
```

TN = 688 , TP = 679 , FN = 45 , FP = 53

:XGBoost

```
#hyperparameters grid
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'n_estimators': [50, 100],
    'max_depth': [None, 3, 5],
    'min_samples_split': [2, 5],  # XGBoost's min_child_weight equivalent
    'subsample': [0.8, 0.9, 1.0],
    'max_features': [0.8, 0.9,  1.0]  # Equivalent to colsample_bytree in XGBoost
}

grid_search = GridSearchCV(estimator=gb_model, param_grid=param_grid, cv=3,
                           n_jobs=-1, verbose=2, scoring='accuracy')
```

ارزیابی:

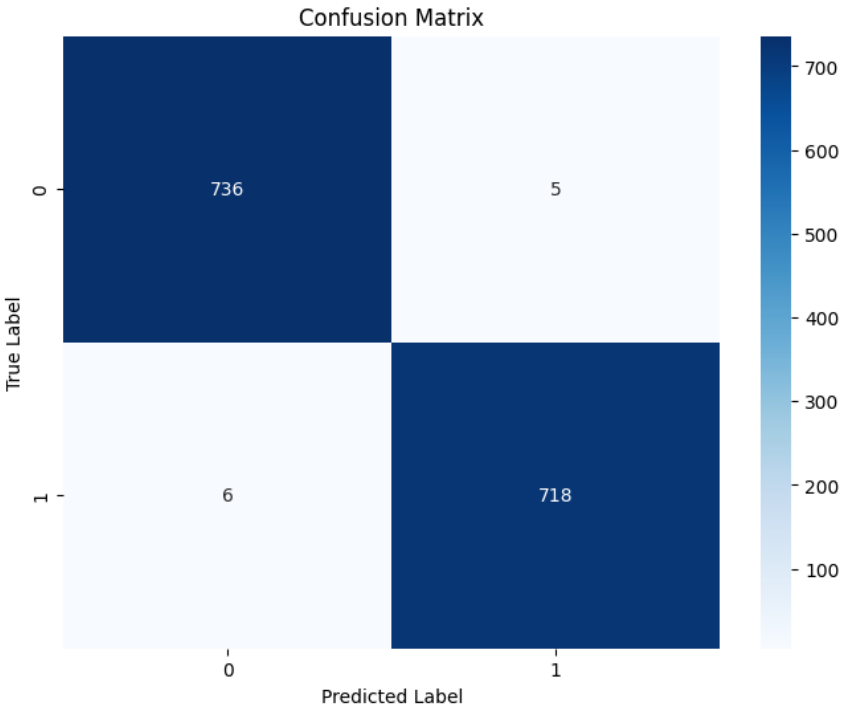



```
Recall: 99.1713%

Precision: 99.3084%

F1-Score: 0.9924

Accuracy: 99.25%

Precision (micro): 0.9925
Precision (macro): 0.9925
Precision (weighted): 0.9925

Recall (micro): 0.9925
Recall (macro): 0.9925
Recall (weighted): 0.9925

F1-Score (micro): 0.9925
F1-Score (macro): 0.9925
F1-Score (weighted): 0.9925
```

TN = 736 , TP = 718 , FN = 6 , FP = 5

:Boosting From Scratch

ارزیابی:




```
Recall: 99.1713%

Precision: 99.3084%

F1-Score: 0.9924

Accuracy: 99.25%

Precision (micro): 0.9925
Precision (macro): 0.9925
Precision (weighted): 0.9925

Recall (micro): 0.9925
Recall (macro): 0.9925
Recall (weighted): 0.9925

F1-Score (micro): 0.9925
F1-Score (macro): 0.9925
F1-Score (weighted): 0.9925
```

TN = 736 , TP = 718 , FN = 6 , FP = 5

دقیقا مشابه حالت قبلی شد!