Hotel Booking Cancel Predict_PPT

강다영

CONTENTS

1. 기획의도

2. 데이터 전처리
&
EDA

3. 모델 구축

4. Confusion Matrix

호텔 예약 취소 (Hotel Booking Cancel)

CANCELLATION RATE BY RESERVATION VALUE

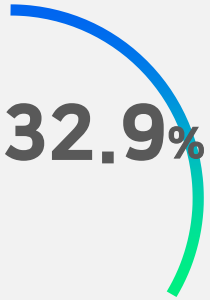
Percentage of on-the-books revenue cancelled before arrival in Europe

	2014	2015	2016	2017	2018	Change
Booking Group	43.4%	43.8%	48.2%	50.9%	49.8%	6.4
Expedia Group	20.0%	25.0%	25.8%	24.7%	26.1%	6.1
Hotelbeds Group	33.2%	37.8%	40.3%	38.3%	37.6%	4.4
HRS Group	58.5%	51.7%	55.2%	59.4%	66.0%	7.5
Other OTAs	13.7%	15.2%	27.0%	24.4%	24.3%	10.6
Other Wholesalers	31.2%	30.3%	34.6%	33.8%	32.8%	1.6
Website Direct	15.4%	17.7%	18.0%	18.4%	18.2%	2.8
AVERAGE	**32.5%**	**34.8%**	**39.6%**	**41.3%**	**39.6%**	**7.1**

Yearly average percentage of on-the-books revenue cancelled prior to guest arrival from a sample of 680 D-EDGE clients in Europe.

D-EDGE, Hospitality Solutions www.d-edge.com

2014-2018 예약 취소 비율(Booking Cancel Percentage in 2014-2018)

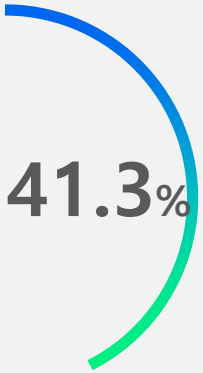


2014

The Average

Cancelation Rate

32.9%

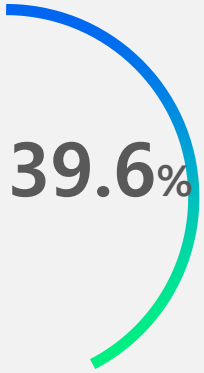


41.3%

2017

The Average

Cancelation Rate



2018

The Average

Cancelation Rate

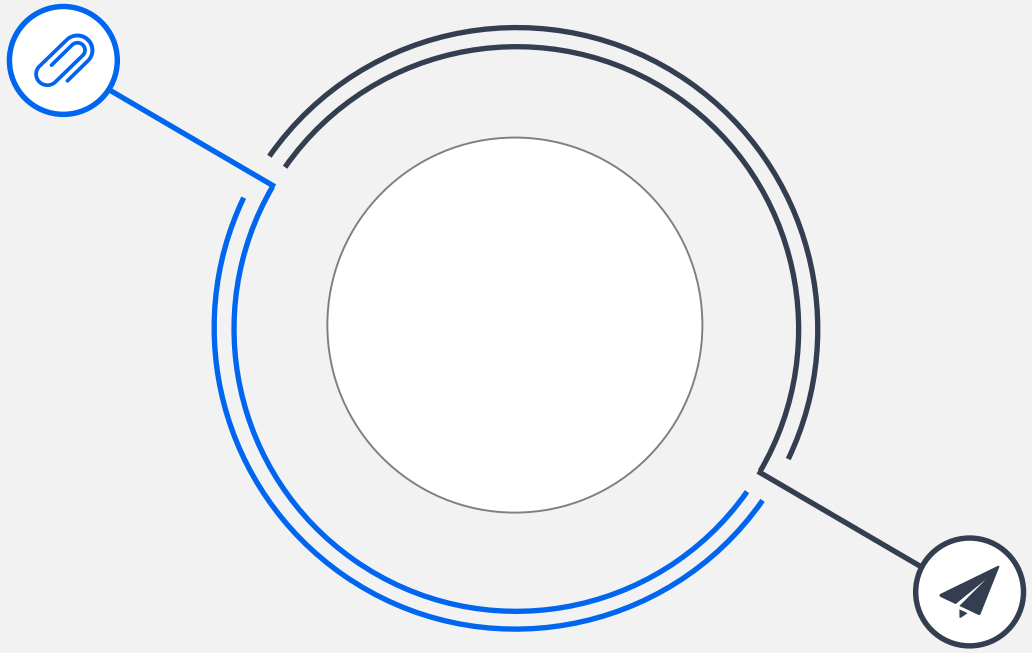
39.6%

예약취소와 노쇼 비율(BOOKING CANCEL & NO-SHOW PERCENTAGE)

40%

BOOKING CANCEL

예약 취소 비율 약 40%



15%

NO-SHOW

노쇼 비율 약 15%

(no-show : 고객이 예약장소에 나타나지 않은 상태)

호텔의 고민거리 = '예약취소'(Hotel's troubles)

<호텔의 오랜 고민?>

-> 예약 취소와 노쇼로 손실발생 -> 손실 줄이기 위한 '초과예약(overbooking)' 시행 -> if 객실 부족? -> 고객 피해 -> 호텔 이미지 손실

∴ 반복(손해 증가) >> 예약 취소 '예측' 필요

데이터 전처리 과정

데이터 전처리 요약.

A

데이터 불러오기

```
h1 = pd.read_csv('./H1.csv')
h1.head()
```

	IsCanceled	LeadTime	ArrivalDateYear	ArrivalDateMonth	ArrivalDateWeekNumber	ArrivalDateDayOfMonth	StaysInWeekendNights	StaysInWeekNights	Adult
0	0	342	2015	July	27	1	0	0	
1	0	737	2015	July	27	1	0	0	
2	0	7	2015	July	27	1	0	1	
3	0	13	2015	July	27	1	0	1	
4	0	14	2015	July	27	1	0	2	

```
h2 = pd.read_csv('./H2.csv')
h2.head()
```

	IsCanceled	LeadTime	ArrivalDateYear	ArrivalDateMonth	ArrivalDateWeekNumber	ArrivalDateDayOfMonth	StaysInWeekendNights	StaysInWeekNights	Adult
0	0	6	2015	July	27	1	0	2	
1	1	88	2015	July	27	1	0	4	
2	1	65	2015	July	27	1	0	4	
3	1	92	2015	July	27	1	2	4	
4	1	100	2015	July	27	2	0	2	

H1,H2 데이터 불러오기

리조트호텔 & 도시호텔 데이터

B

Data Cleaning

```
# 이름 정리(정규식으로 불필요한 문자 제거)
def camel_to_snake(name):
    name = re.sub('(.)([A-Z][a-z]+)', r'W1_W2', name)
    return re.sub('([a-z0-9])([A-Z])', r'W1_W2', name).lower()

# 공백제거
features = ['Meal', 'ReservedRoomType', 'AssignedRoomType', 'DepositType', 'Agent', 'Company']
for feature in features:
    df[feature] = df[feature].str.strip()

# 결측치 찾기
df.isnull().sum()
```

```
IsCanceled                      0
LeadTime                        0
ArrivalDateYear                 0
ArrivalDateMonth                0
ArrivalDateWeekNumber           0
ArrivalDateDayOfMonth           0
StaysInWeekendNights            0
StaysInWeekNights               0
Adults                          0
Children                        4
Babies                          0
Meal                            0
Country                       488
MarketSegment                   0
DistributionChannel             0
IsRepeatedGuest                 0
PreviousCancellations           0
PreviousBookingsNotCanceled     0
```

Data Cleaning

결측치 및 이상치 제거, 정리

C

날짜 정리

```
#도착일 년, 월, 일 분리되어 있는 데이터를 yyyy-mm-dd형태로 바꿔서 새로운 열(함침) 만들기
df['ArrivalDateFull'] = df['ArrivalDateYear'].astype(str) + "-" + df['ArrivalDateMonth'].map({'January':1, 'February':2, 'March':3, 'Apr
df['ArrivalDateFull'] = pd.to_datetime(df['ArrivalDateFull'], format="%Y-%m-%d")
```

```
# reservation_status_date는 고객이 예약을 변경하거나 체크아웃한 시간을 나타낸 것(도착 예정일 혹은 숙박 시작일(arriaval_date와 다를 수 있
# 도착예정일(arrival_date_full)과 마지막예약상태날짜(reservation_status_date)의 차이가 의미하는 것?
# >> 취소를 했거나, 두 날짜 사이에 취소하지 않고 호텔에 머문 시간임
```

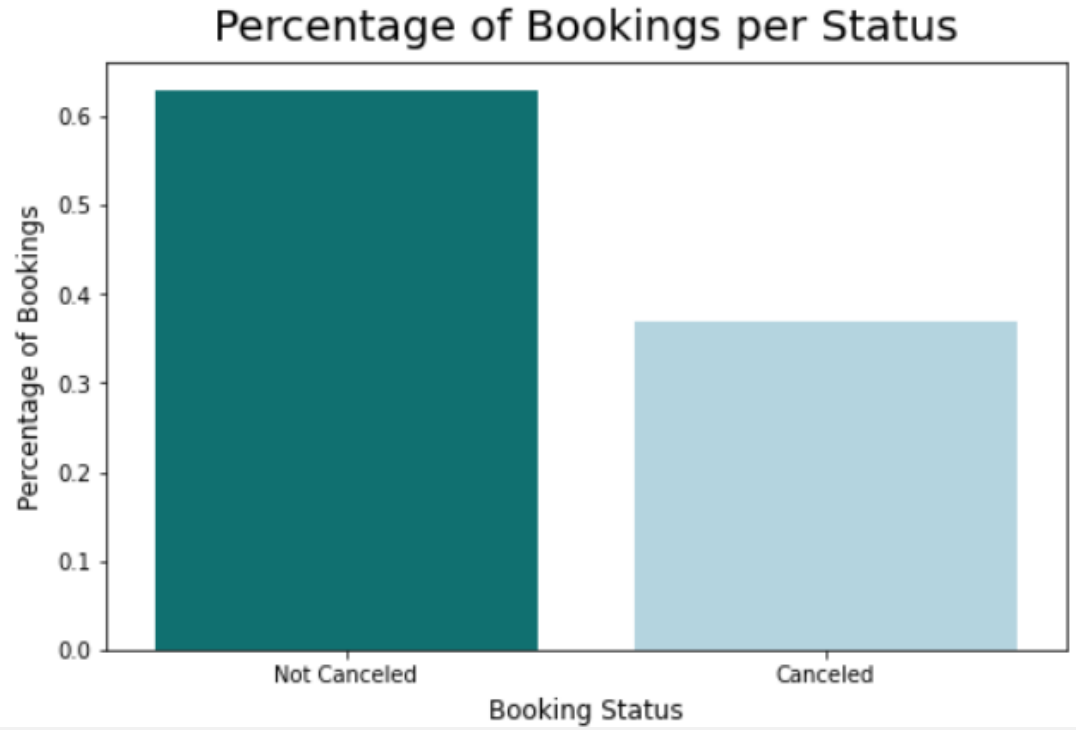
```
df['status_minus_arrival_date'] = np.abs(df['ArrivalDateFull'] - df['ReservationStatusDate']).astype(str)

def format_lenght(date):
    return date[0]
df['status_minus_arrival_date'] = df['status_minus_arrival_date'].map(format_lenght).astype(int)
```

날짜 정리

분리된 연,월,일을 합친 새로운 column 생성

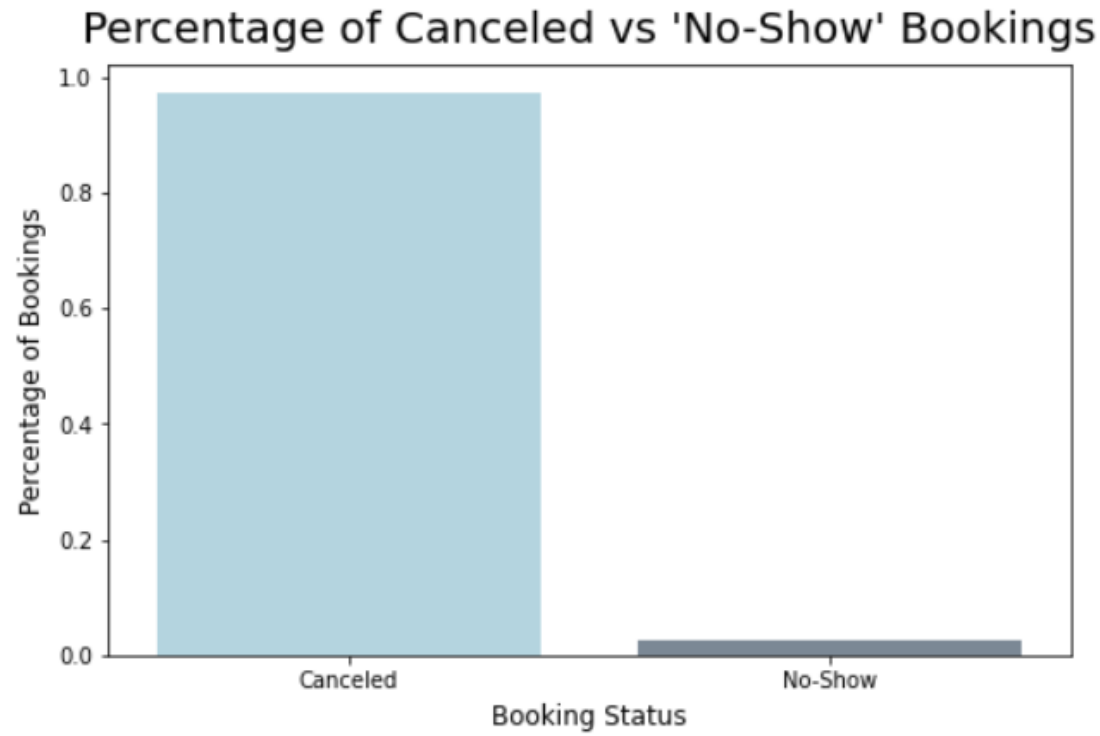
예약과 취소 비율(Booking & Cancel percentage)

```
# 취소 o VS 취소 x
plt.figure(figsize=(8,5))
plt.title("Percentage of Bookings per Status", fontsize = 20, pad = 10)
sns.barplot(x=df['IsCanceled'].unique(), y=df['IsCanceled'].value_counts(normalize=True), palette=['teal', 'lightblue'])
plt.xlabel("Booking Status", fontsize = 12, labelpad = 5)
plt.ylabel("Percentage of Bookings", fontsize = 12, labelpad = 5)
plt.xticks(ticks=[0, 1], labels=['Not Canceled', 'Canceled']);
```



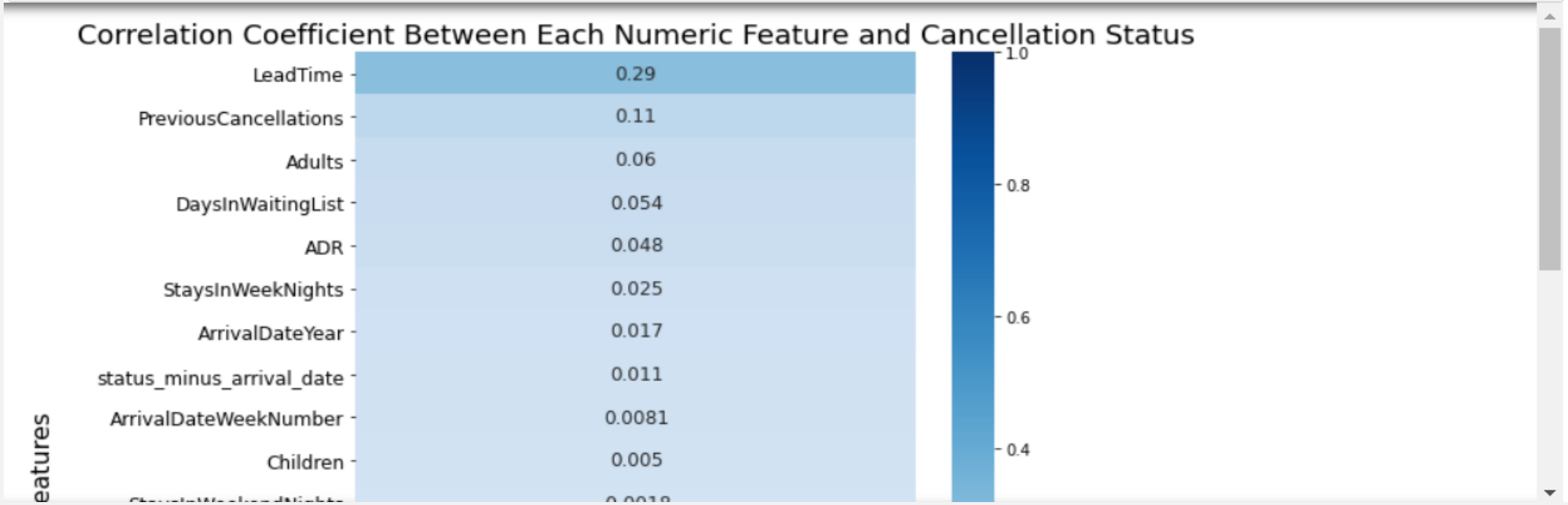
취소와 노쇼의 비율(Cancel & No-show percentage)

```
# 취소 o VS 노쇼
plt.figure(figsize=(8,5))
plt.title("Percentage of Canceled vs 'No-Show' Bookings", fontsize = 20, pad = 10)
sns.barplot(x=df[df['IsCanceled']==1]['ReservationStatus'].unique(), y=df[df['IsCanceled']==1]['ReservationStatus'].value_counts(normali
plt.xlabel("Booking Status", fontsize = 12, labelpad = 5)
plt.ylabel("Percentage of Bookings", fontsize = 12, labelpad = 5);
```



취소와 특성들 사이에 상관관계 파악(Correlation Coefficient)

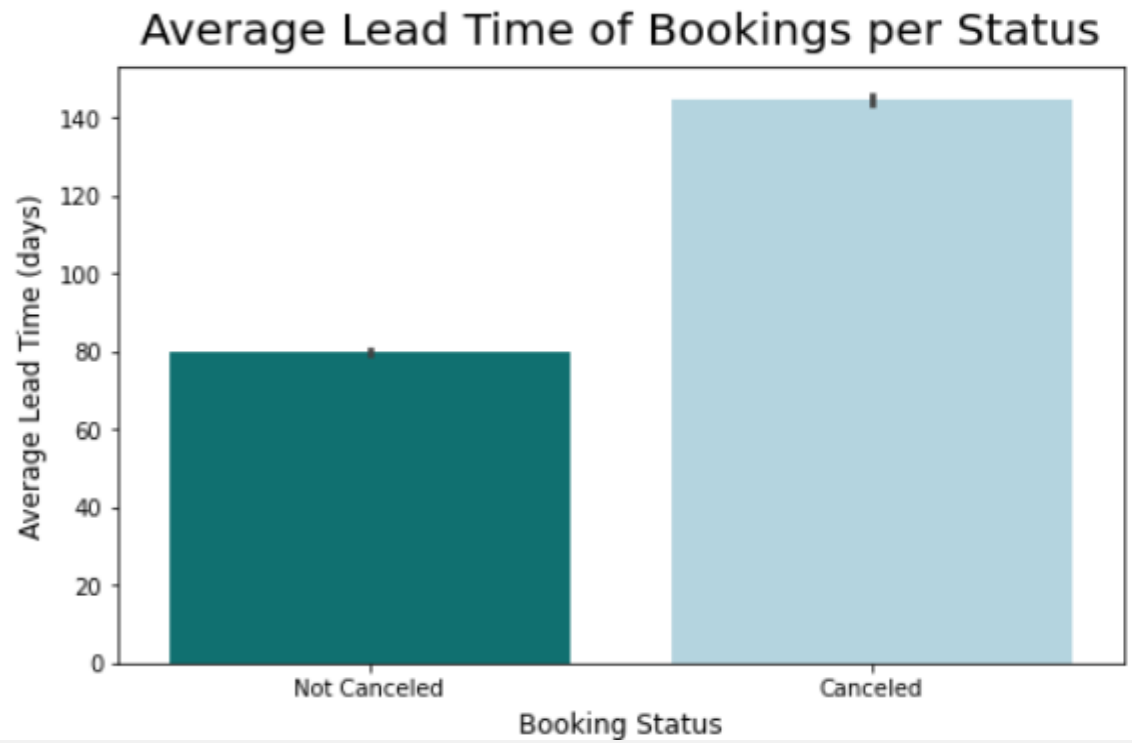
```
# 취소와 특성들 간 상관관계 파악
fig = plt.figure(figsize=(8,10))
ax = sns.heatmap(df.corr()[['IsCanceled']].sort_values('IsCanceled', ascending=False), annot = True, annot_kws = {"size":12}, cmap='Blu
ax.set_title('Correlation Coefficient Between Each Numeric Feature and Cancellation Status', fontsize=18)
ax.set_xlabel('Features', fontsize = 16)
ax.set_ylabel('Features', fontsize = 16)
ax.tick_params(axis = "both", labelsize = 12);
y_min, y_max = ax.get_ylim()
ax.set_ylim(top=y_max+1);
```



상관관계 높은 Features

2-1) Lead Time

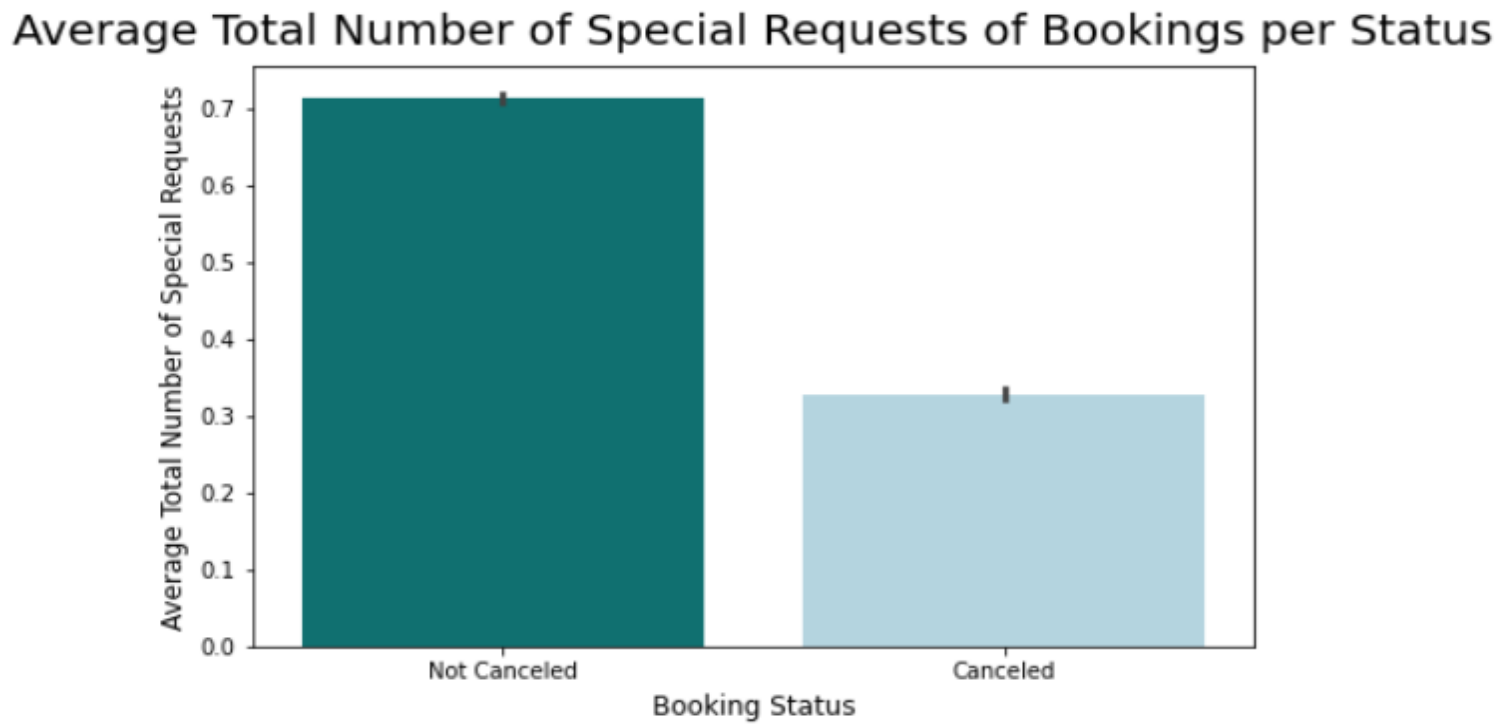
```
#리드타임과 예약 관계 파악
plt.figure(figsize=(8,5))
plt.title("Average Lead Time of Bookings per Status", fontsize = 20, pad = 10)
sns.barplot(x=df['IsCanceled'], y=df['LeadTime'], palette=['teal', 'lightblue'])
plt.xlabel("Booking Status", fontsize = 12, labelpad = 5)
plt.ylabel("Average Lead Time (days)", fontsize = 12, labelpad = 5)
plt.xticks(ticks=[0, 1], labels=['Not Canceled', 'Canceled']);
```



상관관계 높은 Features

2-2) Total Of Special Requests

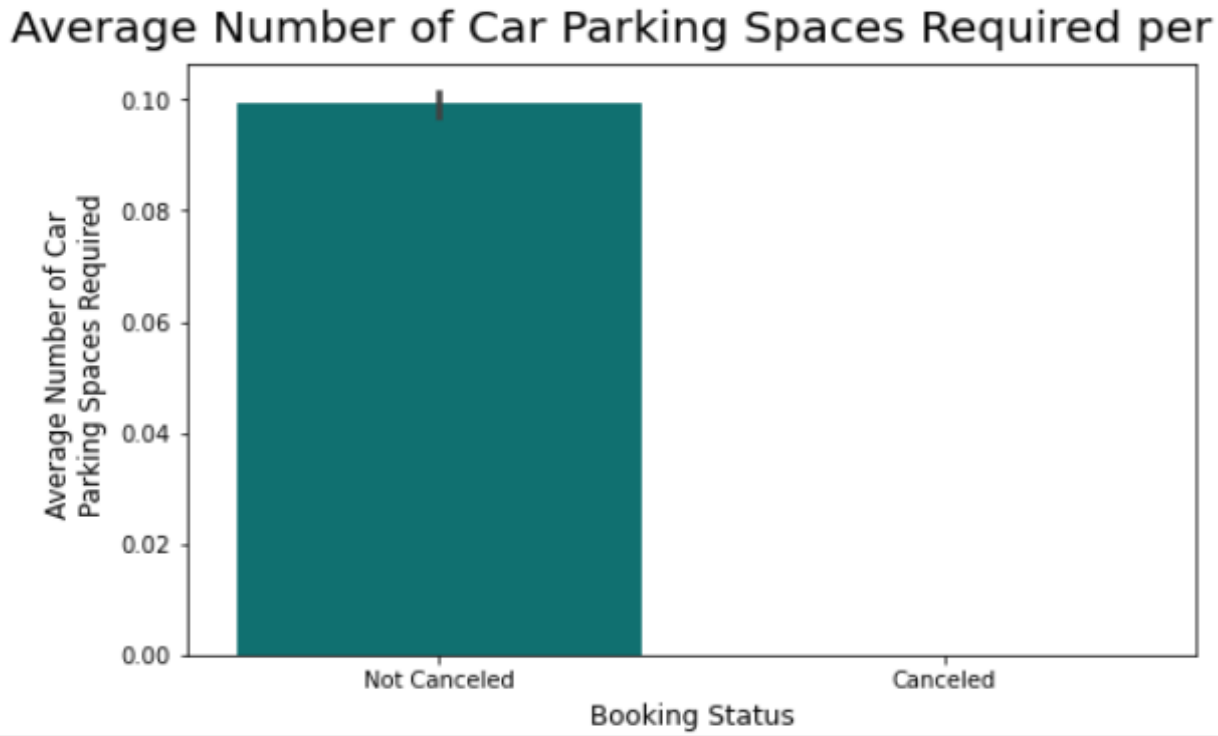
```
# 특별요청과 예약 관계
plt.figure(figsize=(8,5))
plt.title("Average Total Number of Special Requests of Bookings per Status", fontsize = 20, pad = 10)
sns.barplot(x=df['IsCanceled'], y=df['TotalOfSpecialRequests'], palette=['teal', 'lightblue'])
plt.xlabel("Booking Status", fontsize = 12, labelpad = 5)
plt.ylabel("Average Total Number of Special Requests", fontsize = 12, labelpad = 5)
plt.xticks(ticks=[0, 1], labels=['Not Canceled', 'Canceled']);
```



상관관계 높은 Features

2-3) Parking space

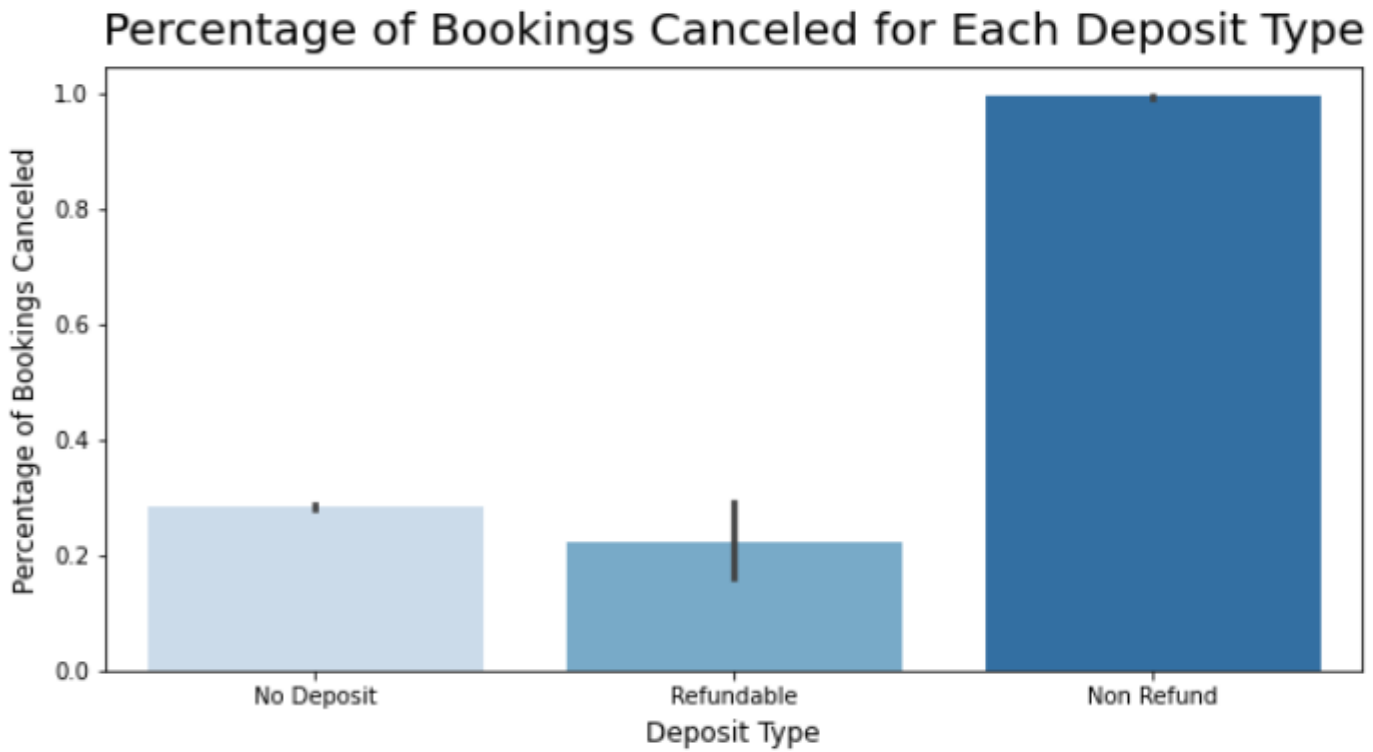
```
# 주차장과 예약 관계
plt.figure(figsize=(8,5))
plt.title("Average Number of Car Parking Spaces Required per Status", fontsize = 20, pad = 10)
sns.barplot(x=df['IsCanceled'], y=df['RequiredCarParkingSpaces'], palette=['teal', 'lightblue'])
plt.xlabel("Booking Status", fontsize = 12, labelpad = 5)
plt.ylabel("Average Number of Car ₩n Parking Spaces Required", fontsize = 12, labelpad = 5)
plt.xticks(ticks=[0, 1], labels=['Not Canceled', 'Canceled']);
```



상관관계 높은 Features

2-4) Deposit type

```
# 보증금 유형과 취소 관계
plt.figure(figsize=(10,5))
plt.title("Percentage of Bookings Canceled for Each Deposit Type", fontsize = 20, pad = 10)
sns.barplot(x=df['DepositType'], y=df['IsCanceled'], palette='Blues')
plt.xlabel("Deposit Type", fontsize = 12, labelpad = 5)
plt.ylabel("Percentage of Bookings Canceled", fontsize = 12, labelpad = 5);
```



Dummy Classifier

Dummy Classifier

```
df = pd.get_dummies(df, columns=['ArrivalDateMonth', 'Meal', 'MarketSegment', 'DistributionChannel', 'ReservedRoomType', 'AssignedRoomTyp
```

```
X = df.drop(columns=['IsCanceled', 'ReservationStatus', 'Agent', 'Company', 'Country', 'ReservationStatusDate', 'ArrivalDateFull'])
y = df['IsCanceled']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=RANDOM_STATE, stratify=y, test_size=0.15)
```

```
dummy = DummyClassifier(strategy='most_frequent')
dummy.fit(X_train,y_train)
```

```
DummyClassifier(strategy='most_frequent')
```

```
print(f'Baseline Training Score : {dummy.score(X_train,y_train)}')
print(f'Baseline Testing Score : {dummy.score(X_test,y_test)}')

# 예약 취소 하지 않는 정확도 약63%
```

```
Baseline Training Score : 0.6296044462839236
Baseline Testing Score : 0.6296068796068796
```

Logistic Regression

로지스틱 회귀

```
lg_params = {
    'penalty': ['l1'],
    'C': [3.5],
    'max_iter': [300]
}

# Perform Grid Search
lg_gs = GridSearchCV(LogisticRegression(solver='liblinear', random_state=RANDOM_STATE),
                     lg_params,
                     cv = 5,
                     scoring = 'accuracy')
lg = lg_gs.fit(X_train, y_train)
```

```
print(f'Best Training Accuracy: {lg.score(X_train, y_train)}')
print(f'Best Testing Accuracy: {lg.score(X_test, y_test)}')
print(f'Cross-val-score: {cross_val_score(lg.best_estimator_, X, y, cv=StratifiedKFold(shuffle=True)).mean()}')
```

```
Best Training Accuracy: 0.8119493880446993
Best Testing Accuracy: 0.8111458566004021
Cross-val-score: 0.8117953834253775
```

```
# 취소여부 제외하고 x가 y에 영향을 미치는 것인가를 확인하는 것
```

Decision Tree

decision tree

```
dt_params = {
    'max_depth':[None],
    'max_features' : [0.7],
    'min_samples_split': [25],
    'min_samples_leaf': [1]
}

# Perform Grid Search
dt_gs = GridSearchCV(DecisionTreeClassifier(random_state=RANDOM_STATE),
                     dt_params,
                     cv = 5,
                     scoring = 'accuracy')
dt = dt_gs.fit(X_train, y_train)
```

```
print(f'Best Training Score : {dt.score(X_train,y_train)}')
print(f'Best Testing Score : {dt.score(X_test,y_test)}')
print(f'cross-val-score : {cross_val_score(dt.best_estimator_,X,y,cv=StratifiedKFold(shuffle=True)).mean()}')
```

```
Best Training Score : 0.9619621987031672
Best Testing Score : 0.9415345097163279
cross-val-score : 0.9388789668529842
```

Random Forest

Random Forest

```
rf_params = {
    'n_estimators': [150],
    'max_features': [50],
    'max_depth': [13]
}

# Perform Grid Search
rf_gs = GridSearchCV(RandomForestClassifier(random_state=RANDOM_STATE),
                     rf_params,
                     cv = 5,
                     scoring = 'accuracy')
rf = rf_gs.fit(X_train, y_train)
```

```
print(f'Best Training Accuracy: {rf.score(X_train, y_train)}')
print(f'Best Testing Accuracy: {rf.score(X_test, y_test)}')
print(f'Cross-val-score: {cross_val_score(rf.best_estimator_, X, y, cv=StratifiedKFold(shuffle=True)).mean()}')
```

```
Best Training Accuracy: 0.9227615837915607
Best Testing Accuracy: 0.9198682153227608
Cross-val-score: 0.9196304490729693
```

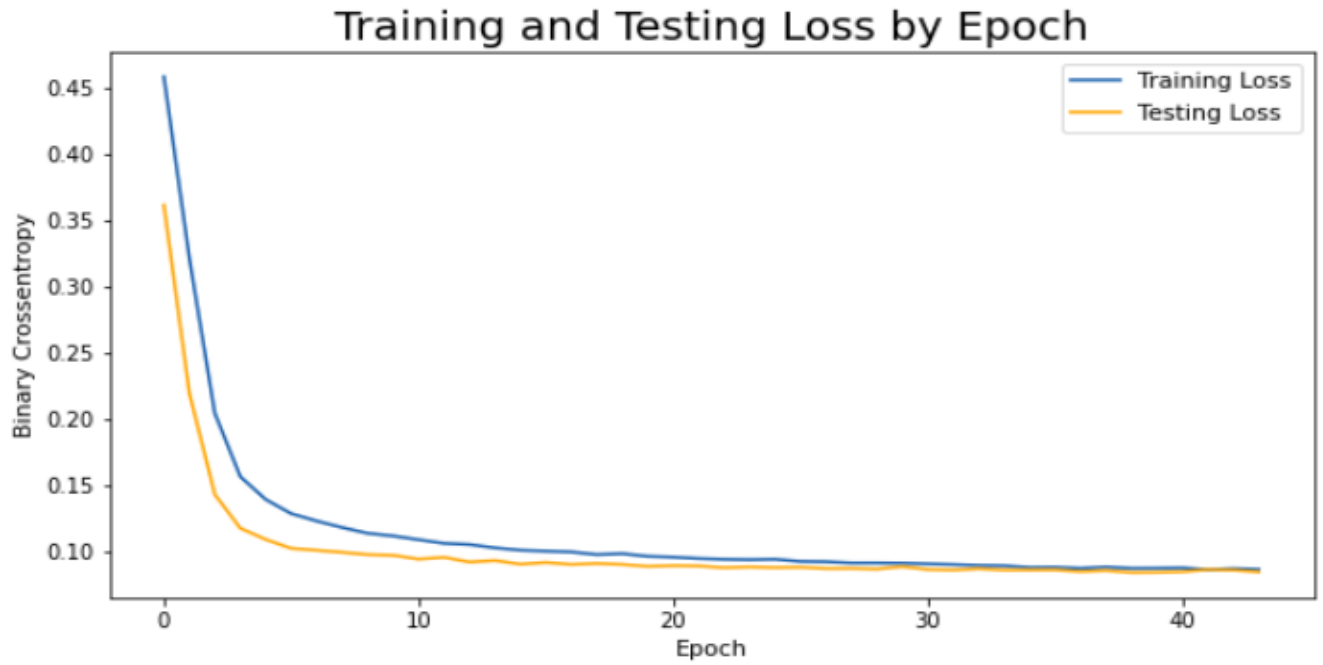
Neural Networks

Neural Networks

```
ss = StandardScaler()
X_train_sc = ss.fit_transform(X_train)
X_test_sc = ss.transform(X_test)

# Creating our model's structure
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(77,)))
model.add(Dropout(0.18))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.15))
model.add(Dense(1, activation='sigmoid'))
es = EarlyStopping(monitor='val_loss', patience=5)

# Compiling the model
model.compile(loss='bce',
              optimizer='adam',
              metrics=['binary_accuracy'])

# Fitting the model
history = model.fit(X_train_sc,
                    y_train,
                    batch_size = 256,
                    validation_data =(X_test_sc, y_test),
                    epochs = 500,
                    verbose = 0,
                    callbacks=[es])
```

Neural Networks(Training & Testing Loss)

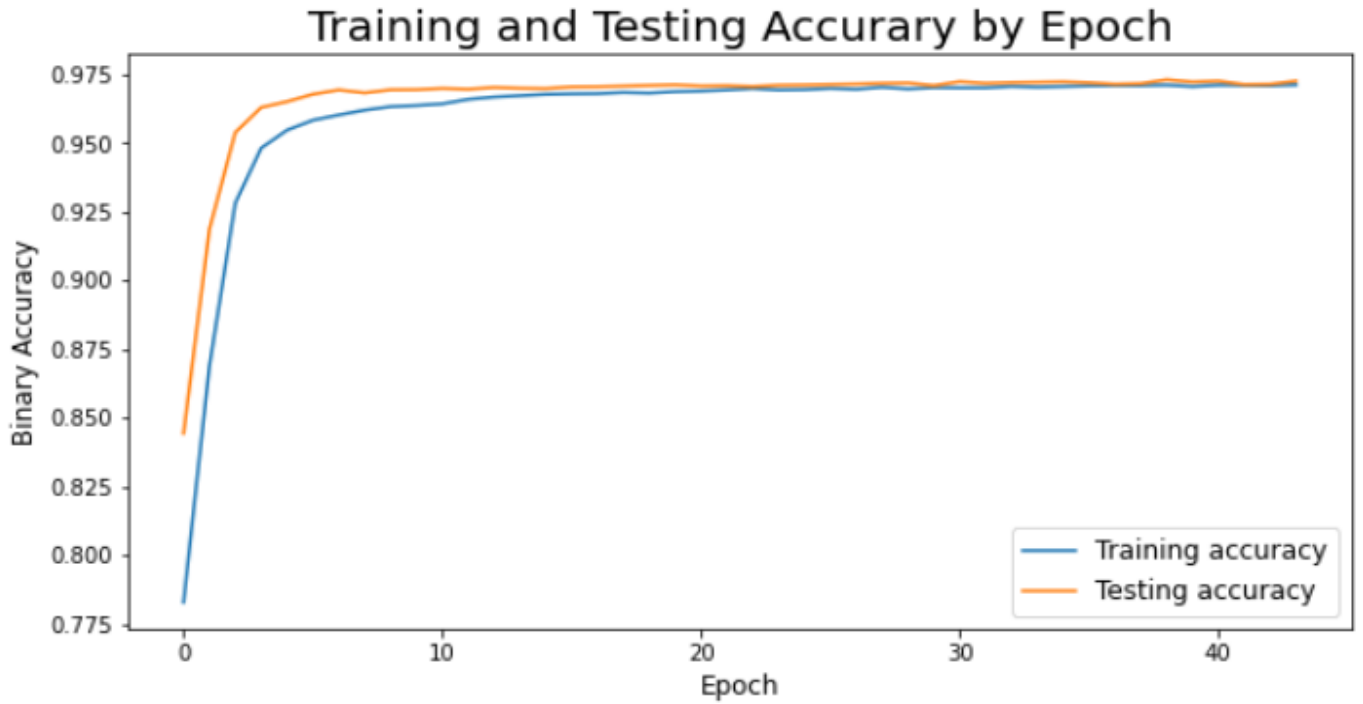
```
train_loss = history.history['loss']
test_loss = history.history['val_loss']

# Visualizing our training and testing loss by epoch
plt.figure(figsize=(10, 5))
plt.plot(train_loss, label='Training Loss', color='#185fad')
plt.plot(test_loss, label='Testing Loss', color='orange')
plt.title('Training and Testing Loss by Epoch', fontsize = 20)
plt.xlabel('Epoch', fontsize = 11)
plt.ylabel('Binary Crossentropy', fontsize = 11)
plt.legend(fontsize = 11);
```



Neural Networks(Training & Testing Accuracy)

```
plt.figure(figsize=(10, 5))
plt.plot(history.history['binary_accuracy'], label='Training accuracy')
plt.plot(history.history['val_binary_accuracy'], label='Testing accuracy')
plt.title('Training and Testing Accurary by Epoch', fontsize = 20)
plt.xlabel('Epoch', fontsize = 12)
plt.ylabel('Binary Accuracy', fontsize = 12)
plt.legend(fontsize = 12)
```

```
<matplotlib.legend.Legend at 0x2d2eece8340>
```



Neural Networks(Training & Testing Accuracy)

```
train_score = model.evaluate(X_train_sc,
                             y_train,
                             verbose=1)
test_score = model.evaluate(X_test_sc,
                            y_test,
                            verbose=1)
labels = model.metrics_names

print('')
print(f'Training Accuracy: {train_score[1]}')
print(f'Testing Accuracy: {test_score[1]}')
```

```
3172/3172 [==============================] - 14s 4ms/step - loss: 0.0744 - binary_accuracy: 0.9749
560/560 [==============================] - 2s 4ms/step - loss: 0.0850 - binary_accuracy: 0.9726

Training Accuracy: 0.9748713970184326
Testing Accuracy: 0.9725821018218994
```

Confusion Matrix

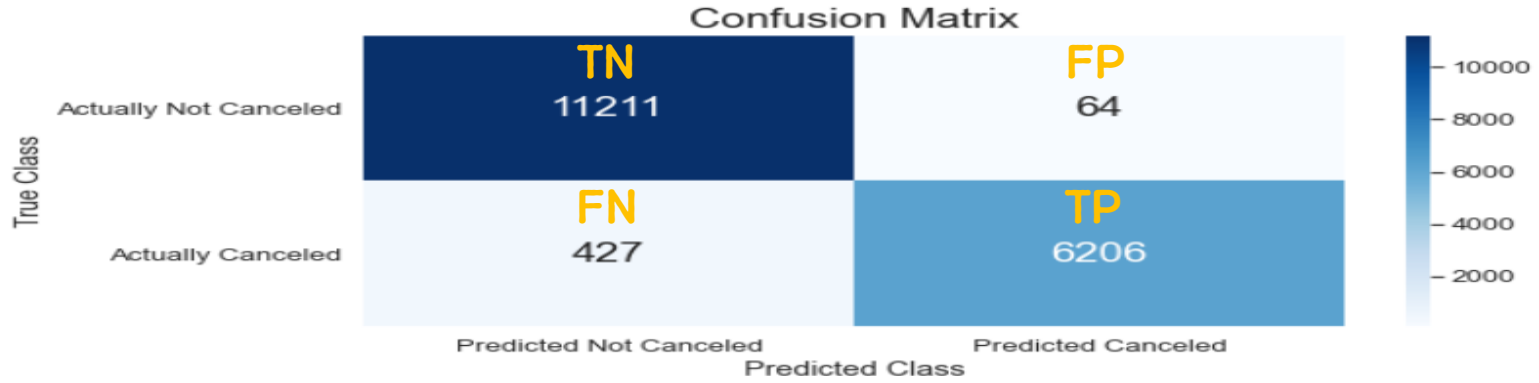
confusion matrix

```
nn_predictions = model.predict_classes(X_test_sc)

# Creating confusion matrix
cm = confusion_matrix(y_test, nn_predictions)

# putting the matrix a dataframe form
cm_df = pd.DataFrame(cm, index=['Actually Not Canceled', 'Actually Canceled'],
                     columns=['Predicted Not Canceled', 'Predicted Canceled'])
```

```
C:₩Users₩687₩anaconda3₩lib₩site-packages₩tensorflow₩python₩keras₩engine₩sequential.py:450: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

```
sns.set(font_scale=1.2)
plt.figure(figsize=(10,4))

sns.heatmap(cm, annot=True, fmt='g', cmap="Blues",xticklabels=cm_df.columns, yticklabels=cm_df.index, annot_kws={"size": 20})
plt.title("Confusion Matrix", size=20)
plt.xlabel('Predicted Class')
plt.ylabel('True Class');
```

```
sns.set(font_scale=1.2)
plt.figure(figsize=(10,4))

sns.heatmap(cm, annot=True, fmt='g', cmap="Blues",xticklabels=cm_df.columns, yticklabels=cm_df.index, annot_kws={"size": 20})
plt.title("Confusion Matrix", size=20)
plt.xlabel('Predicted Class')
plt.ylabel('True Class');
```



Neural Networks(Training & Testing Accuracy)

```
# True Positives:
TP = 6212
# True Negatives:
TN = 11197
# False Posititives:
FP = 78
# False Negatives:
FN = 421
total = 6212+421+11197+78

print(f'Correctly classified: {np.round((TP+TN)/total*100)}%')
print(f'Canceled bookings correctly classified: {np.round(TP/(TP+FN)*100)}%')
print(f'Not canceled bookings correctly classified: {np.round(TN/(TN+FP)*100)}%')
print(f'Bookings predicted canceled that are actually canceled: {np.round(TP/(TP+FP)*100)}%')
print(f'Bookings predicted not canceled that are actually not canceled: {np.round(TN/(TN+FN)*100)}%')
```

```
Correctly classified: 97.0%
Canceled bookings correctly classified: 94.0%
Not canceled bookings correctly classified: 99.0%
Bookings predicted canceled that are actually canceled: 99.0%
Bookings predicted not canceled that are actually not canceled: 96.0%
```

결론 및 대안제시

Neural Network 모델 정확도 가장 높음
: 리드타임, 특별요청사항(주차공간), 환불불가 보증금이 취소에 영향을 미친다는 결론

1. **Lead time** 예약은 체크인 날짜 80일 전부터 오픈하여 취소율을 낮출 수 있음

2. **Non-refund Deposit** 환불불가는 취소 ↑ : 보증금 높이거나 환불불가 제거

3. **Special Requests** 특별 요청 많으면 취소율 낮음, 손님에게 매력적인 요청 제공