Data Science Challange

Predicting Hotel Booking Cancellation

Latar Belakang

Industri perhotelan menghadapi tantangan unik dalam mengelola ketersediaan kamar, memahami perilaku pelanggan, dan meningkatkan efisiensi operasional. Dalam upaya mengatasi ketidakpastian ini, proyek prediksi pemesanan hotel bertujuan memanfaatkan analisis data dan pembelajaran mesin untuk mengidentifikasi pola dan faktor-faktor yang mempengaruhi keputusan pelanggan dalam membatalkan atau melanjutkan pemesanan.

Goals

1. Mengembangkan model prediktif untuk memprediksi kemungkinan pembatalan pemesanan hotel
2. Memberikan rekomendasi operasional kepada hotel berdasarkan temuan dari analisis data

Deskripsi Dataset

- Data ini terdiri dari 31 kolom dan 82,665 baris.
- Dari 31 kolom tersebut, 11 kolom berisi data kategorikal, 19 kolom lainnya berisi data numerikal, dan satu kolom berisi data dengan tipe datetime.
- Variabel Target adalah is_canceled

	hotel	is_canceled	lead_time	arrival_date_year
35414	Las Vegas Marriott Las Vegas, NV	1	28	2017
49887	Newark Liberty International Airport Courtyard...	0	54	2018
64906	Spokane Downtown at the Convention Center Cour...	1	310	2018

Data Cleaning

Missing value

Ada 4 kolom yang mengandung missing value. Untuk menangani nilai yang hilang tersebut, dilakukan langkah-langkah berikut: satu kolom dihapus karena memiliki missing value>90%, satu kolom baru dibuat baru, dan baris yang mengandung nilai yang hilang dihapus.

Incorrect value

Menghapus nilai yang tidak masuk akal

Duplicate value

Menghapus baris yang terduplikat

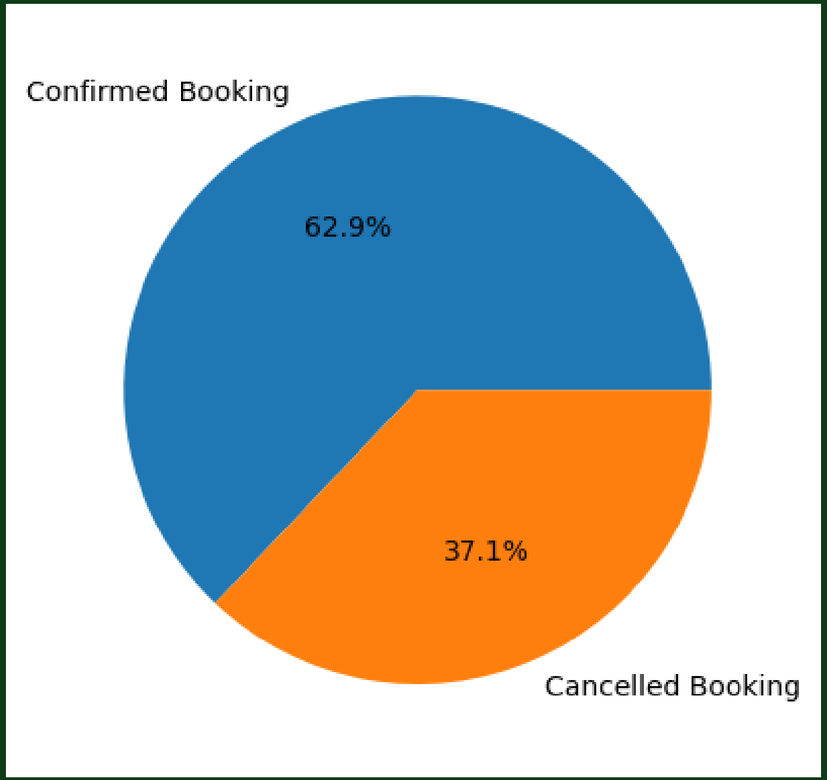
Incorrect data type

Mengubah tipe sesuai dengan kebutuhan

Exploratory Data Analysis

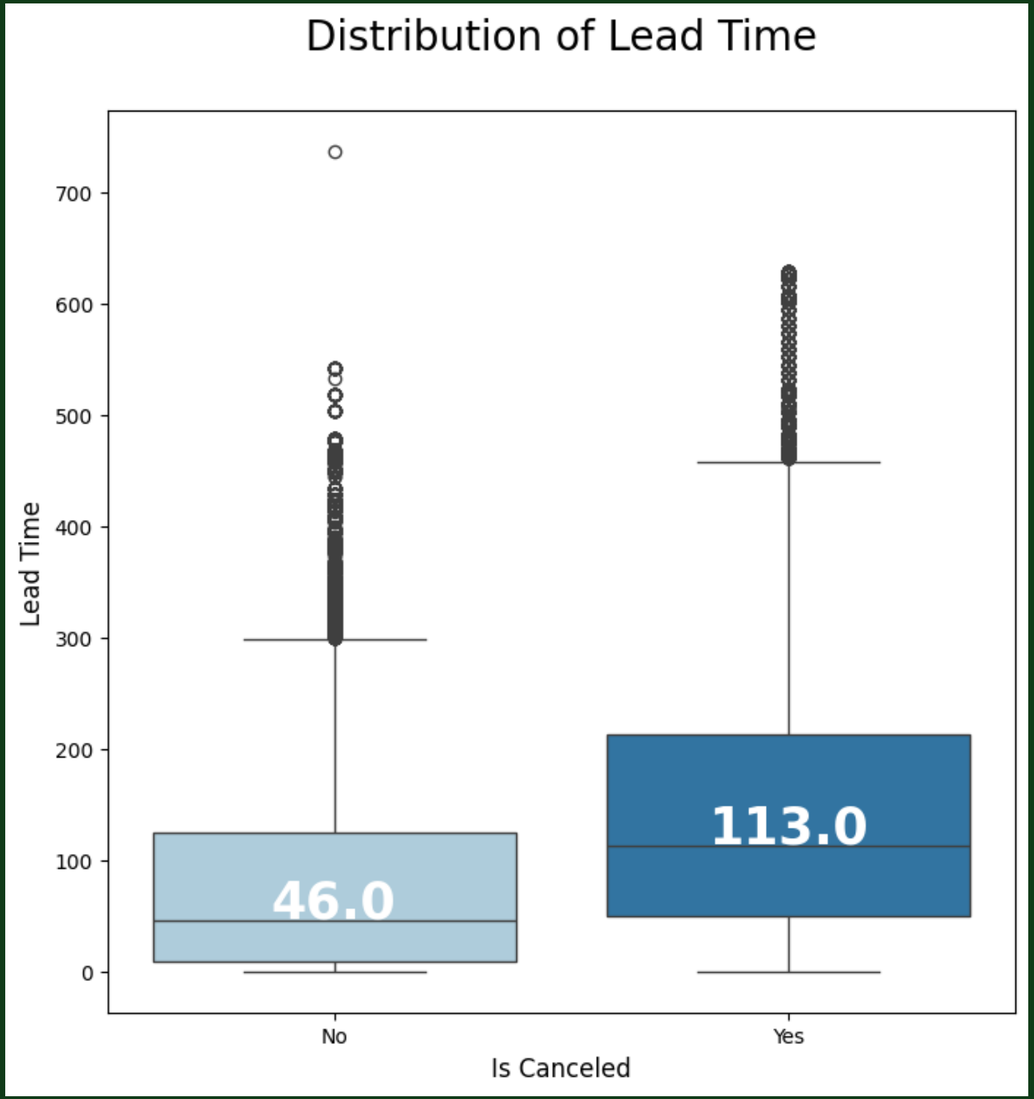
Distribution Label Target

Secara proporsi, perbandingan antar kelas tidak seimbang. Proporsi antar kelas adalah 63% : 37%



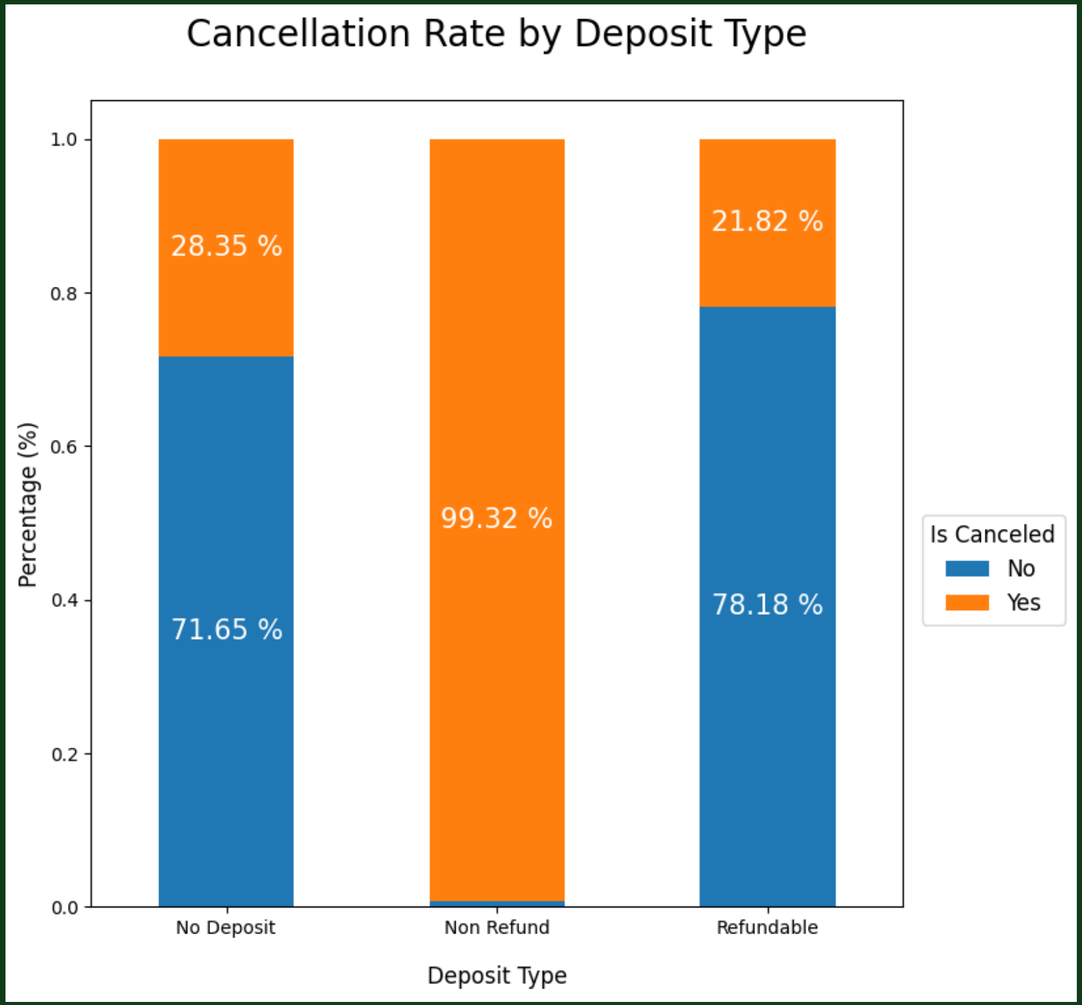
Lead Time dan Cancellation

- Semakin lama waktu tunggu, semakin besar kemungkinan pelanggan akan membatalkan pemesanan hotel
- Masuk akal jika jumlah hari antara saat pemesanan dilakukan dan perkiraan tanggal kedatangan bertambah, pelanggan memiliki lebih banyak waktu untuk membatalkan pemesanan dan lebih banyak waktu untuk keadaan tak terduga yang dapat menggagalkan rencana perjalanan



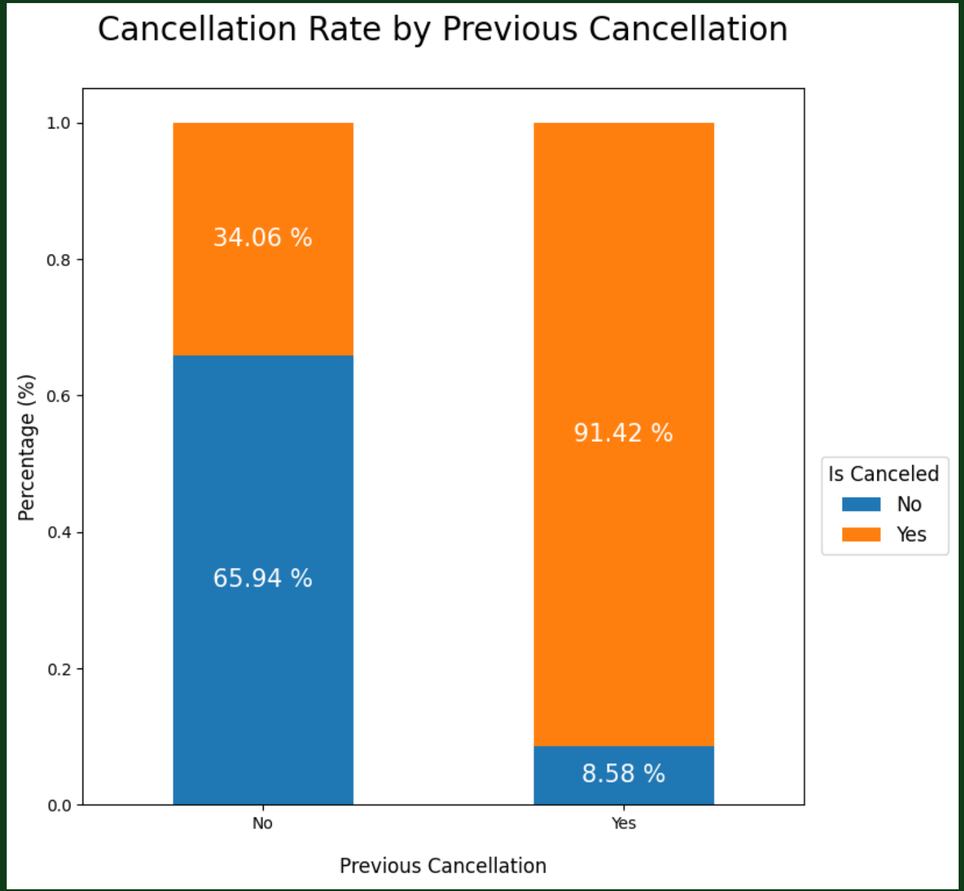
Deposit Type and Cancellation

- Pemesanan dengan deposit non refund memiliki tingkat pembatalan tertinggi, memberikan indikasi bahwa kebijakan deposit dapat mempengaruhi keputusan pembatalan.



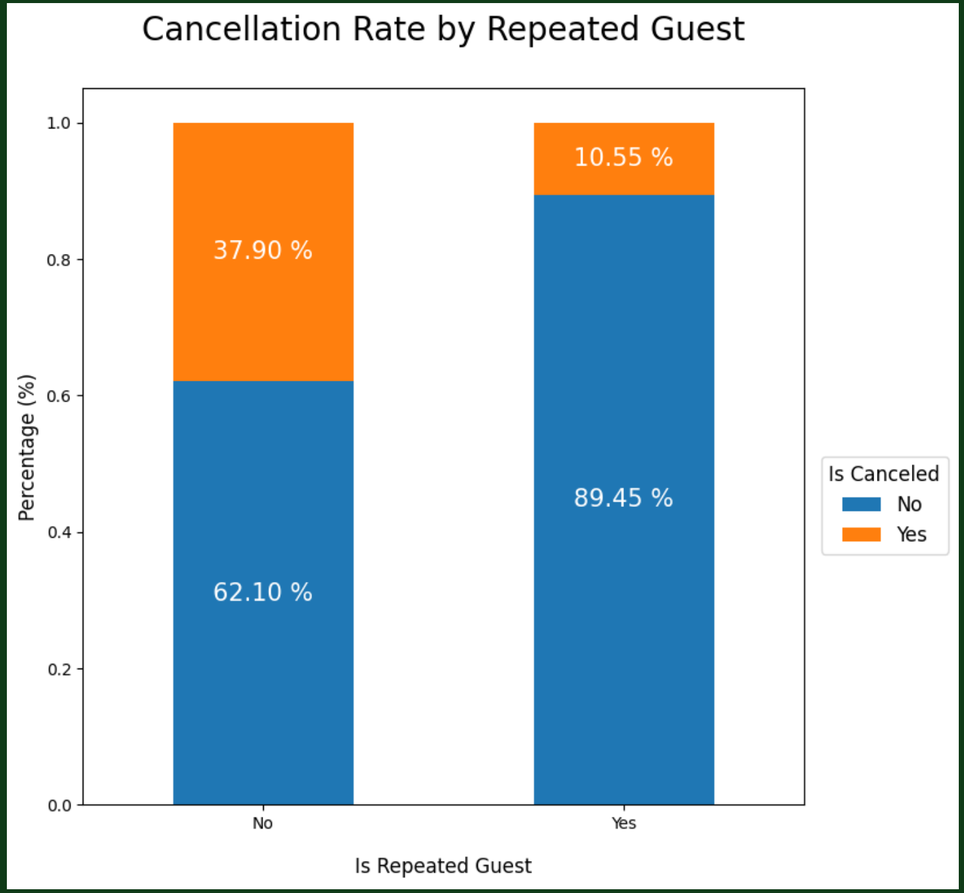
Previous Cancellations dan Cancellation

Previous Cancellations secara signifikan meningkatkan kemungkinan terjadinya pembatalan dimasa yang akan datang.

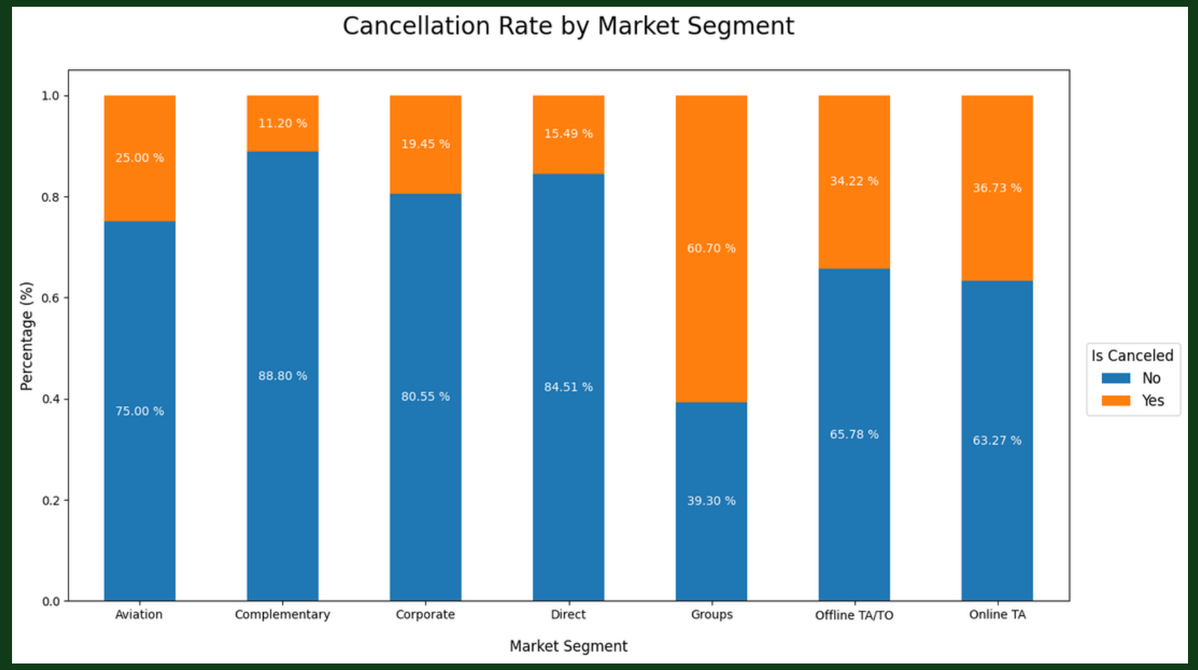


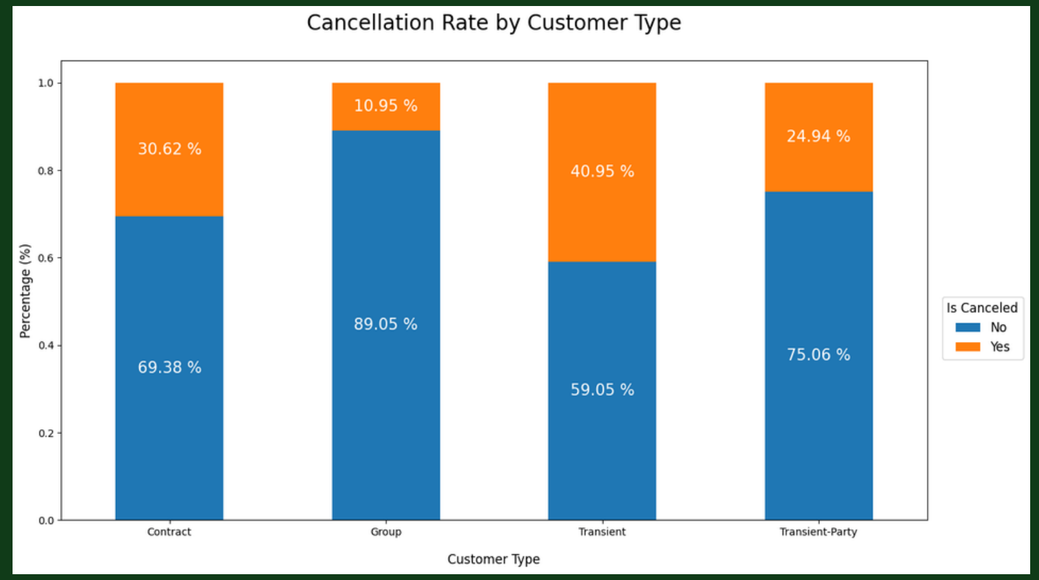
Repeated Guest dan Cancellation

Tamu yang berulang kali datang memiliki kemungkinan lebih kecil untuk membatalkan pemesanan.



Market Segment dan Customer Type

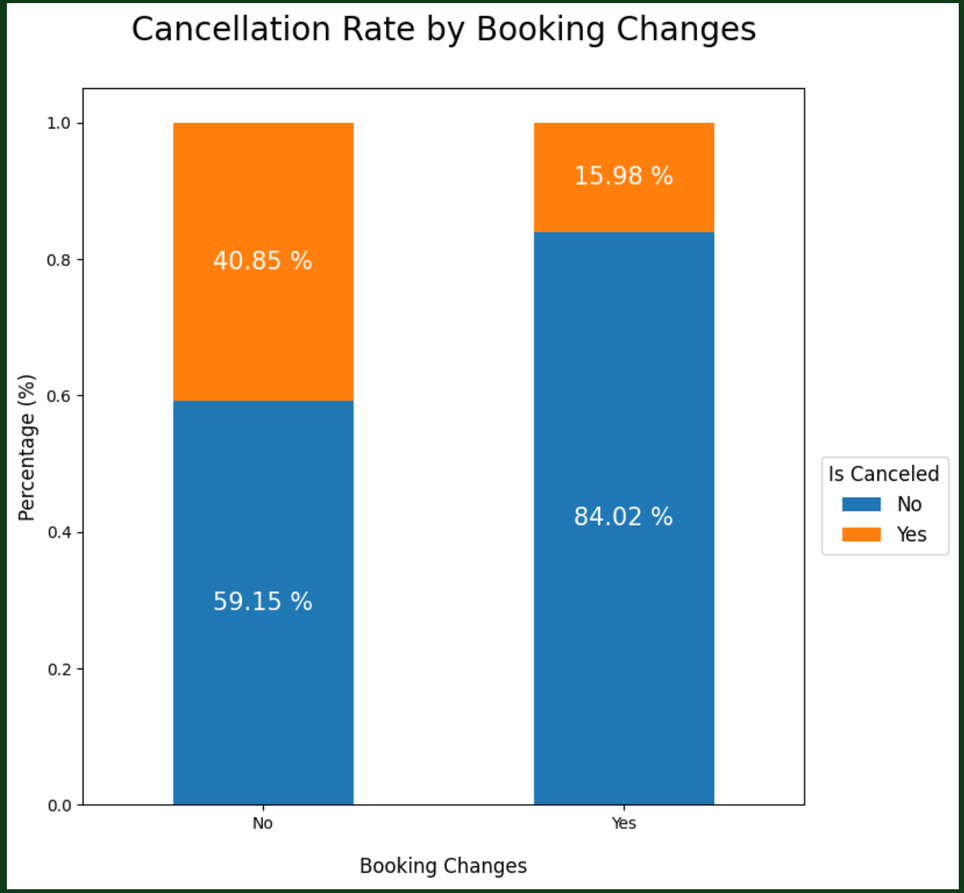




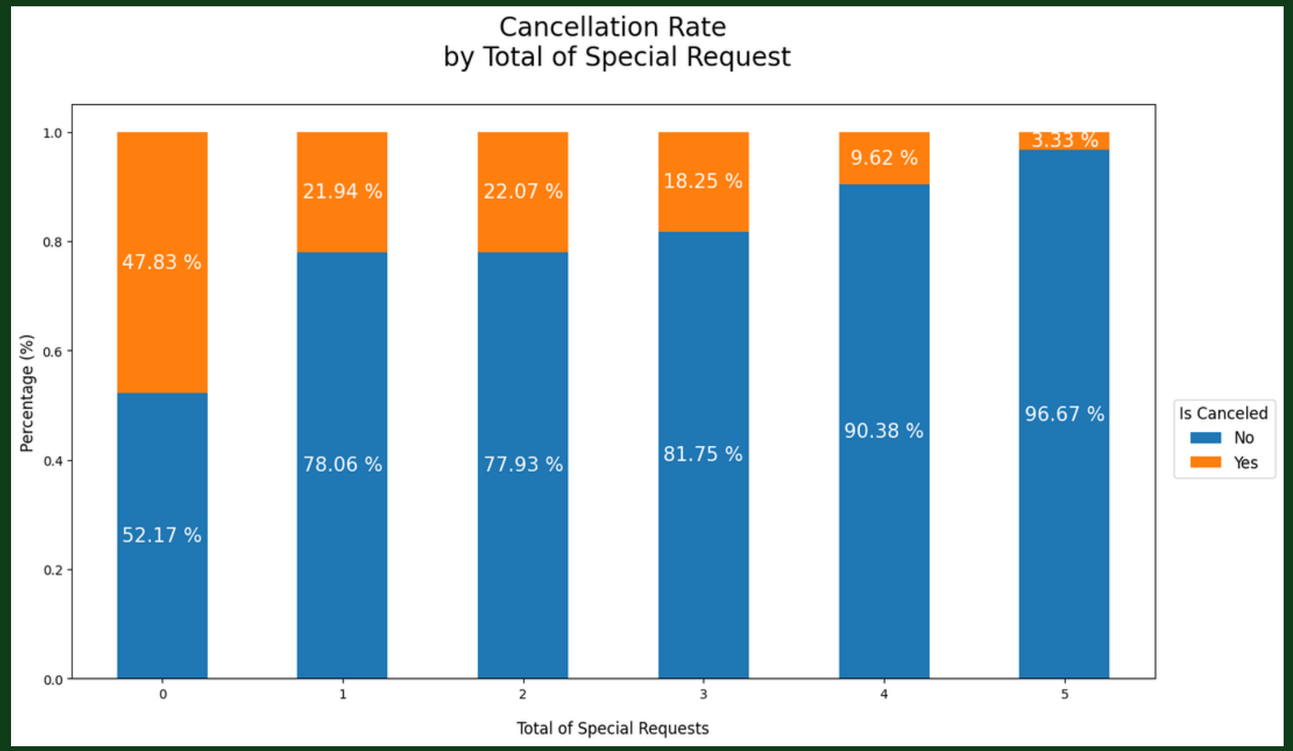
Segmen pasar dan jenis pelanggan tertentu memiliki tingkat pembatalan yang lebih tinggi, menunjukkan pentingnya memahami profil pelanggan untuk mengelola risiko pembatalan.

Booking Changes dan Cancellation

Pelanggan yang melakukan perubahan pada detail pemesanan memiliki tingkat pembatalan yang lebih rendah.

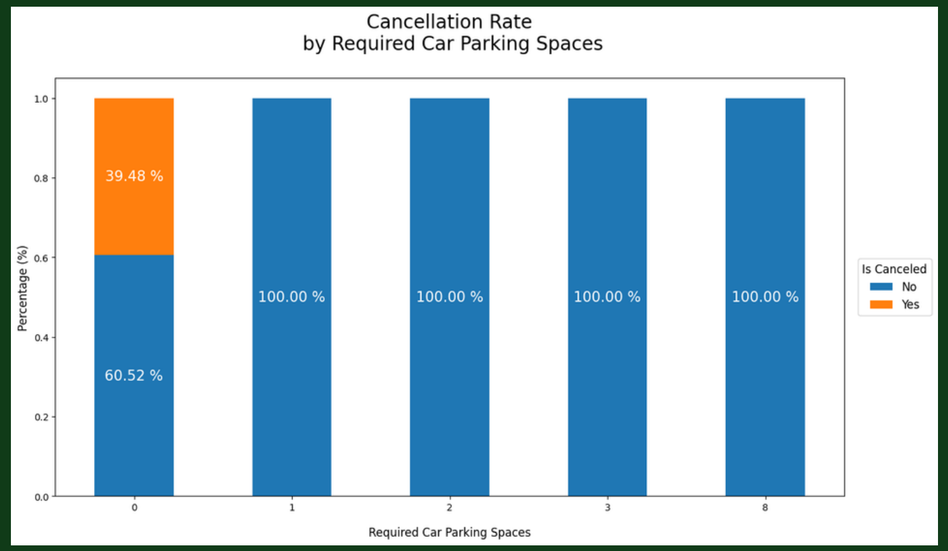


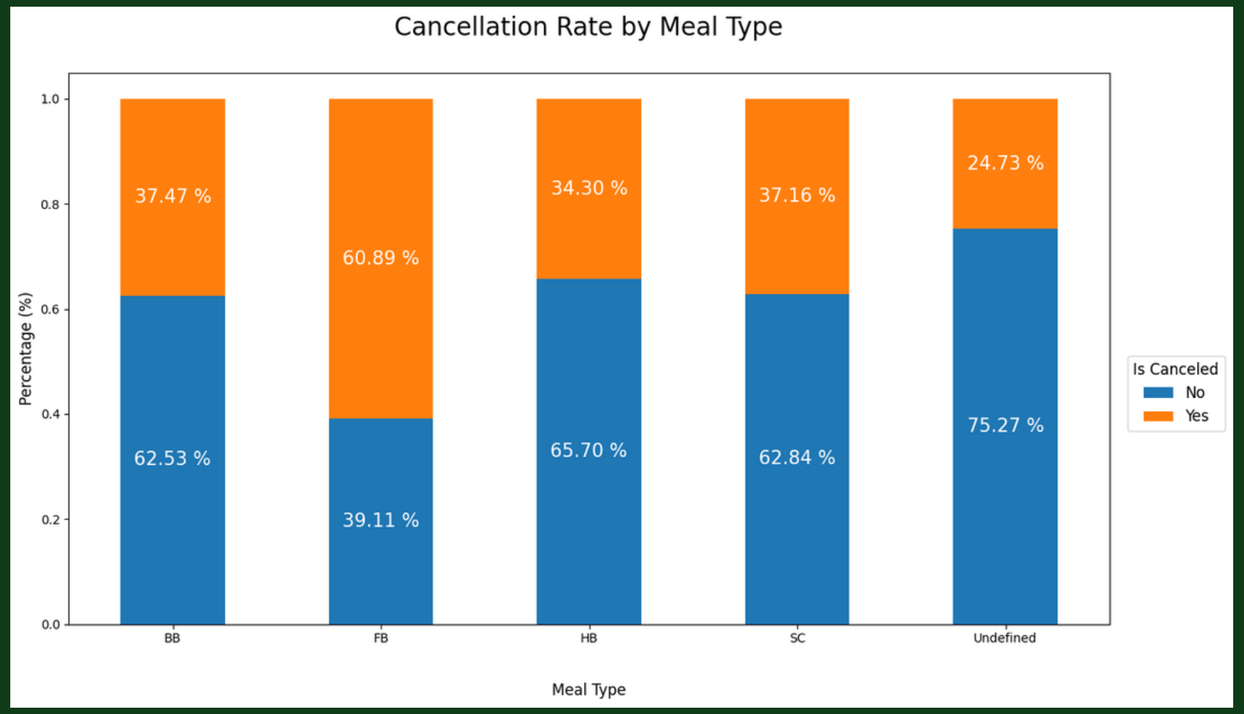
Special Request dan Cancellation:



Pelanggan yang meminta permintaan khusus memiliki tingkat pembatalan yang lebih rendah.

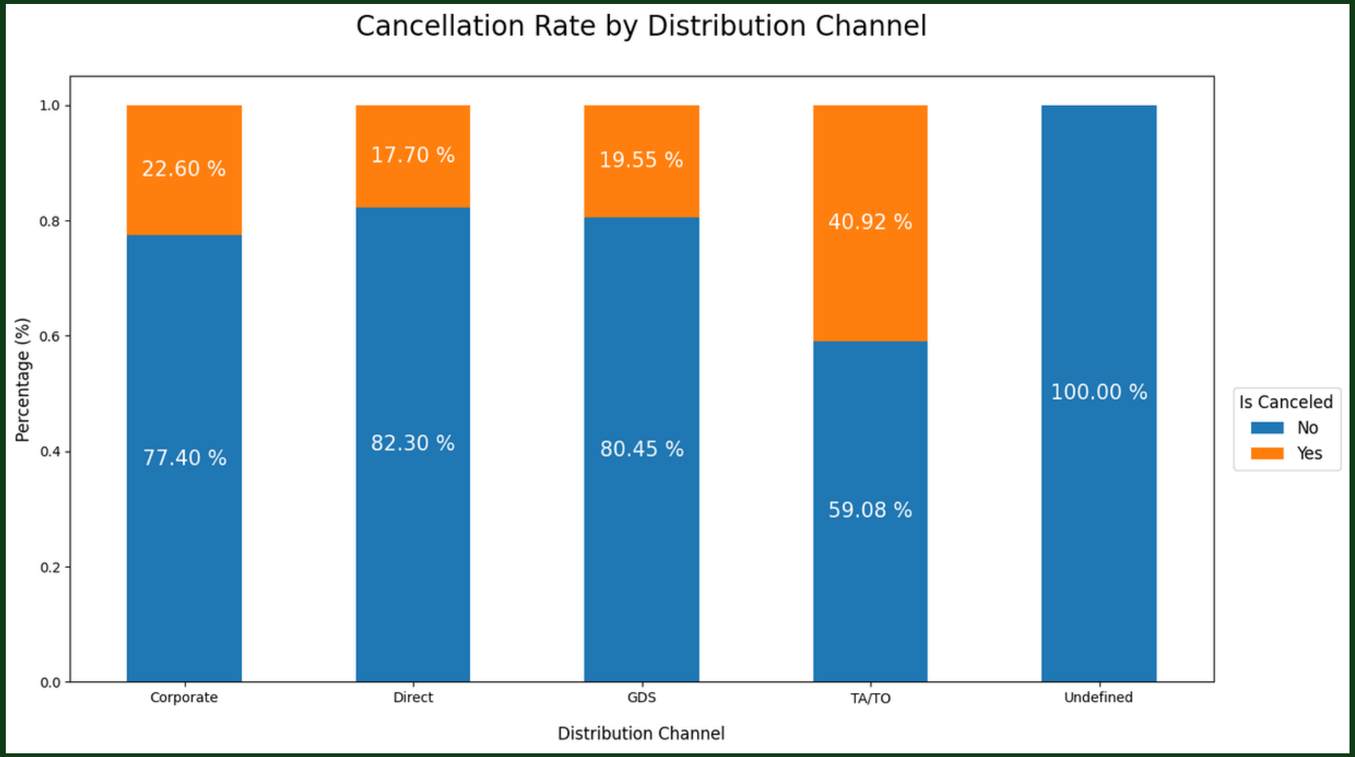
Parking Space dan Meal Type





Permintaan parkir dan jenis makanan dapat memengaruhi keputusan pembatalan.

Distribution Channel dan Cancellation



Saluran distribusi tertentu memiliki tingkat pembatalan yang lebih tinggi, yang dapat mencerminkan kebijakan atau karakteristik pemesanan melalui saluran tersebut.

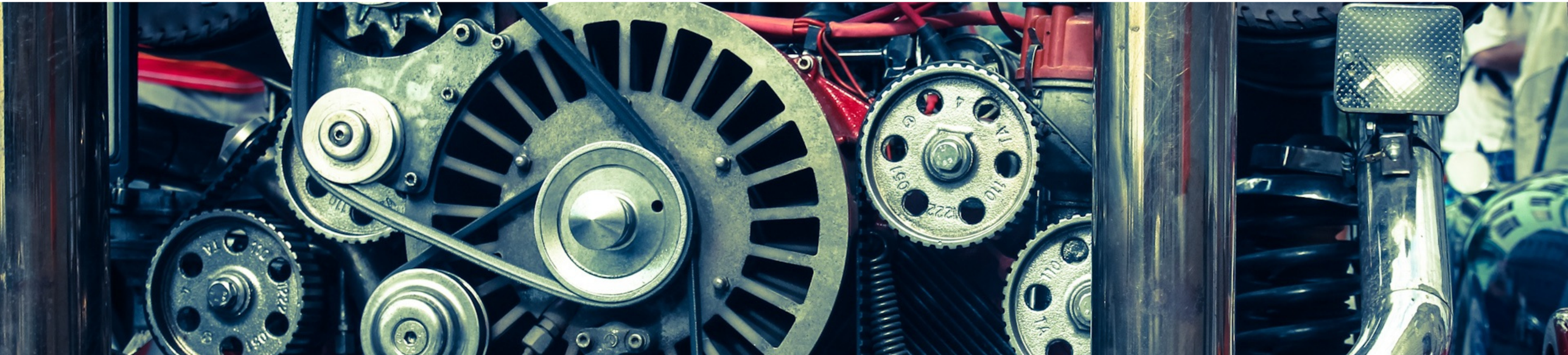
Feature Engineering

Feature Engineering

1. Penambahan fitur-fitur baru:
   - Total Stays
   - Total Guests
   - Total Kids
   - Guest Location
   - cancelrate_byMarket
   - RateSpecialReq_to_TotalGuest
2. Simplifikasi fitur: required_car_parking_spaces -> park_car_req
3. Transformasi Data: Transformasi fitur adr dan lead time dengan Box Cox untuk mengurangi skewness
4. Feature Scaling
5. Encoding Feature Categorical
6. Resampling data dengan Undersampling karena label target proporsinya tidak seimbang

Model Building

Logistc Regression, Decision Tree, Random Forest, XGBoost



Fungsi untuk membangun model

```python
def calcu_eval_all(models, X_train, X_test, y_train, y_test):
    f1_scores, auc_scores, times, fprs, tprs, get_model, fnr = [], [], [], [], [], [], []

    for name, model in models.items():
        fpr, tpr = [], []
        f1 = auc = exectime = 0

        start = time.process_time()
        if name == "XGBoost" or name == "XGBoost Tuned":
            X_train_np = X_train.to_numpy()
            y_train_np = y_train.to_numpy()
            clf = model.fit(X_train_np, y_train_np)
            y_prob = clf.predict_proba(X_test)[:, 1]
            y_pred = clf.predict(X_test)
        else:
            clf = model.fit(X_train, y_train)
            y_prob = clf.predict_proba(X_test)[:, 1]
            y_pred = clf.predict(X_test)
        exectime = round(time.process_time() - start, 4)

        # Evaluate Metrics
        fpr, tpr, _ = roc_curve(y_test, y_prob)
        cm = confusion_matrix(y_test, y_pred)
        f1 = round(f1_score(y_test, y_pred), 4)
        auc = round(roc_auc_score(y_test, y_prob), 4)
        report = classification_report(y_test, y_pred)
        false_negative_rate = cm[1, 0] / (cm[1, 0] + cm[1, 1])

        # Print Evaluation Metrics
        print("==================================================================================\n")
        print(f"Nama: {name}\nFalse negative rate: {false_negative_rate}\nAUC Score: {auc}\nF1-Score: {f1}\nConfusion Matrix:\n{cm}\nReport:\n{report}")
        print("==================================================================================\n")

        f1_scores.append(f1)
        auc_scores.append(auc)
        times.append(exectime)
        fprs.append(fpr)
        tprs.append(tpr)
        get_model.append(clf)
        fnr.append(false_negative_rate)

    return f1_scores, auc_scores, times, fprs, tprs, get_model, fnr
```

Tipe Model

Baseline Model

```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

models = {
    "Logistic Regression": LogisticRegression(),
    "Decisioion Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "XGBoost": XGBClassifier()}
```

✓ 0.0s

Model with hyperparamter

```python
models = {
    "Logistic Regression": LogisticRegression(solver='liblinear', max_iter=1000),
    "Decisioion Tree": DecisionTreeClassifier(),
    "Decisioion Tree Tuned": DecisionTreeClassifier(criterion='gini',
                                                    max_depth=15,
                                                    max_features=None,
                                                    min_samples_leaf=8,
                                                    min_samples_split=8,
                                                    splitter='random'),
    "Random Forest": RandomForestClassifier(),
    "Random Forest Tuned": RandomForestClassifier(n_jobs=-1,
                                                  criterion='entropy',
                                                  max_depth=30,
                                                  max_features='sqrt',
                                                  min_samples_leaf=1,
                                                  min_samples_split= 2,
                                                  bootstrap=True,
                                                  n_estimators=1300),
    "XGBoost": XGBClassifier(),
    "XGBoost Tuned": XGBClassifier(tree_method='auto',
                                   min_child_weight=1,
                                   max_depth=60,
                                   reg_lambda=1.0,
                                   gamma=1.0,
                                   eta=0.20202020202020204,
                                   colsample_bytree=0.30000000000000004,
                                   alpha=0.7000000000000001)}
```

✓ 0.0s

Hyperparameter Tuning

Decision Tree

```
param_dist = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': randint(1, 20),  # Misalnya, jangkauan dari 1 hingga 20
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 20),
    'max_features': ['sqrt', 'log2', None]
}

dt_model = DecisionTreeClassifier()
random_search = RandomizedSearchCV(dt_model, param_distributions=param_dist,
                                   n_iter=50, scoring='accuracy', cv=5,
                                   verbose=5, n_jobs=-1, return_train_score=True)

random_search.fit(X_train, y_train)
print("Best Parameters:", random_search.best_params_)
```

Random Forest

```
rf_para = dict(
    n_estimators = [int(x) for x in np.linspace(start = 100, stop = 2000, num = 20)], #
    bootstrap = [True],
    criterion = ['gini','entropy'],
    max_depth = [int(x) for x in np.linspace(10, 110, num = 11)],
    min_samples_split = [int(x) for x in np.linspace(start = 2, stop = 10, num = 5)],
    min_samples_leaf = [int(x) for x in np.linspace(start = 1, stop = 10, num = 5)],
    max_features = [None, 'sqrt', 'log2'],
    n_jobs = [-1],
)

rf = RandomForestClassifier()
rf_cv = RandomizedSearchCV(rf, param_distributions=rf_para, n_iter=25,
                           scoring='accuracy', cv=5, verbose=4, n_jobs=-1,
                           return_train_score=True)

rf_cv.fit(X_train, y_train)
print("Best parameters (RF): {}".format(rf_cv.best_params_))
```

XGBoost

```
hyperparameters = {
    'max_depth' : [int(x) for x in np.linspace(10, 110, num = 11)],
    'min_child_weight' : [int(x) for x in np.linspace(1, 20, num = 11)],
    'gamma' : [float(x) for x in np.linspace(0, 1, num = 11)],
    'tree_method' : ['auto', 'exact', 'approx', 'hist'],
    'colsample_bytree' : [float(x) for x in np.linspace(0, 1, num = 11)],
    'eta' : [float(x) for x in np.linspace(0, 1, num = 100)],
    'lambda' : [float(x) for x in np.linspace(0, 1, num = 11)],
    'alpha' : [float(x) for x in np.linspace(0, 1, num = 11)]
}

xg = XGBClassifier()
xg_tuned = RandomizedSearchCV(xg, hyperparameters, cv=5, n_iter=25, scoring='accuracy',
                              verbose=3, return_train_score=True)
X_train_np = X_train.to_numpy()
y_train_np = y_train.to_numpy()
xg_tuned.fit(X_train_np,y_train_np)
print("Best paramters (XGBoost): {}".format(xg_tuned.best_params_))
```

Model Evaluation

Model Comparison

Model	F1-Score	F1-Score_B	False negative rate	False negative rate_B	Runtime(s)	Runtime_B(s)
Random Forest Tuned	0.8169	0.8141	0.212527	0.153319	162.2344	91.0625
XGBoost Tuned	0.8135	0.8137	0.217583	0.144675	55.4219	37.8594
Random Forest	0.8130	0.8109	0.215789	0.154624	7.8750	5.6406
XGBoost	0.8071	0.8039	0.223291	0.159843	19.8906	16.2969
Decisioion Tree Tuned	0.7779	0.7748	0.243843	0.185451	0.2656	0.1250
Decisioion Tree	0.7766	0.7679	0.220845	0.177459	0.4375	0.3281
Logistic Regression	0.7293	0.7517	0.336813	0.216441	0.7500	0.5625

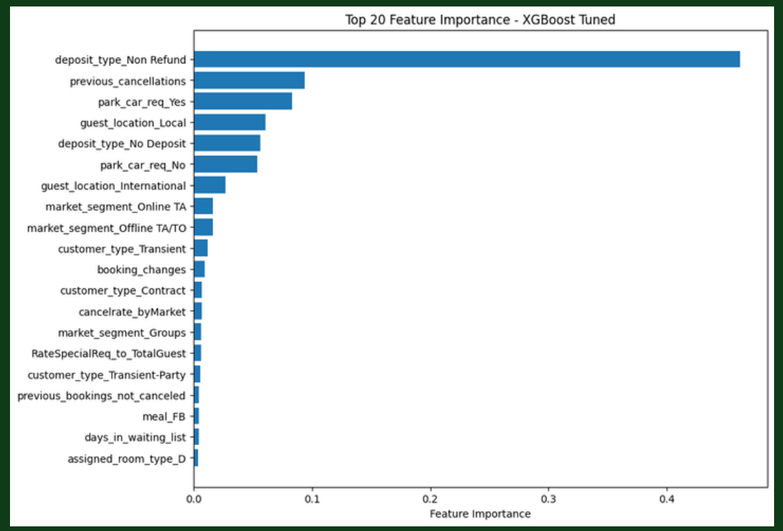
XGBoost Tuned dan Random Forest Tuned, menunjukkan kinerja yang baik dengan nilai F1-Score yang mendekati satu sama lain.

Namun, jika dilihat dari false negatives, XGBoost Tuned memiliki nilai False Negative Rate yang sedikit lebih rendah daripada Random Forest Tuned.

Oleh karena itu, berdasarkan F1-Score dan False Negative Rate, XGBoost Tuned pada resampling menjadi pilihan terbaik.

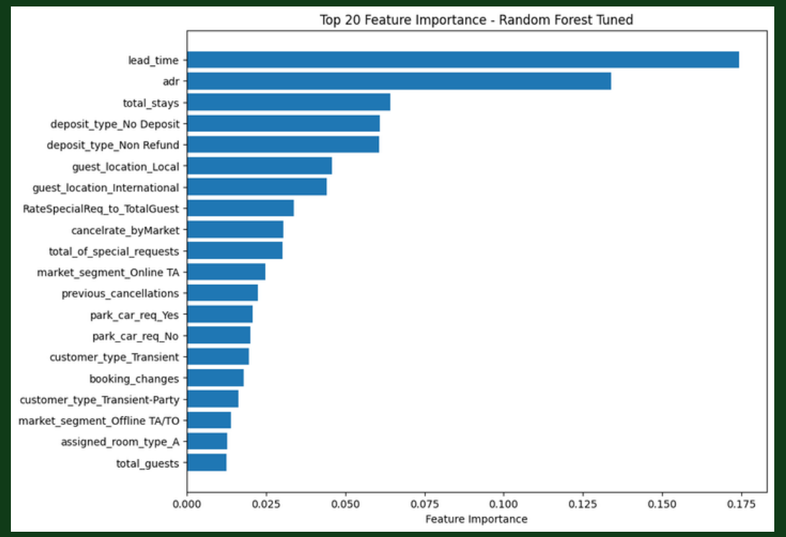
Feature Important

XGBoost



Model XGBoost menekankan pentingnya jenis deposit, riwayat pembatalan sebelumnya, permintaan parkir, dan lokasi tamu dalam membuat prediksi. deposit_type_Non Refund menjadi fitur yang paling penting, menyoroti dampak dari kebijakan yang tidak dapat dikembalikan pada keputusan pembatalan.

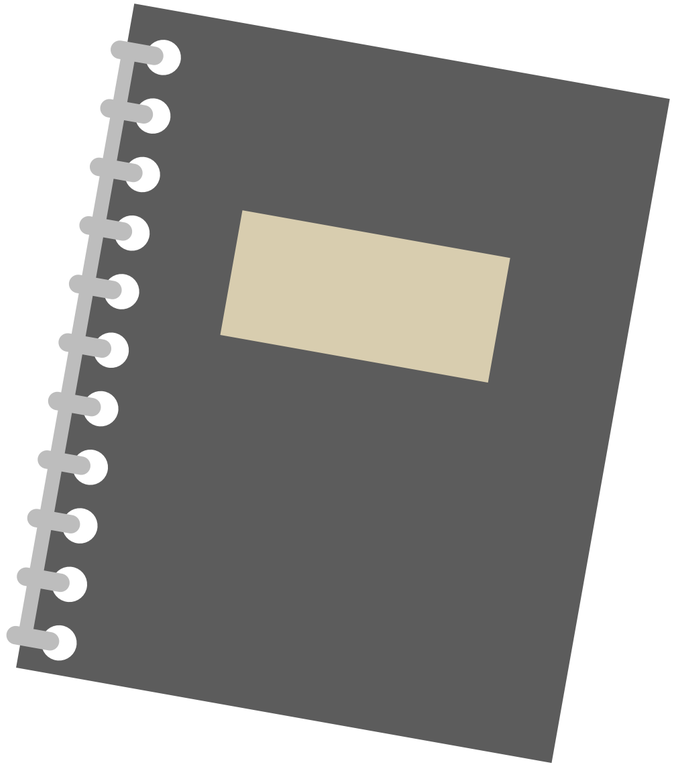
Random Forest



Model Random Forest menyoroti peran waktu antara pemesanan dan kedatangan (Lead Time), tarif harian rata-rata (adr), jumlah total masa inap, dan jenis deposit dalam pembatalan. Lead Time merupakan fitur yang paling penting, menunjukkan pentingnya waktu sebagai faktor prediksi.

Rekomendasi

- Tinjau dan sesuaikan kebijakan deposit untuk meningkatkan fleksibilitas dan meminimalkan hambatan pembatalan.
- Berikan penekanan khusus pada lead time dan ADR. Mungkin ada kebijakan pembatalan yang lebih fleksibel untuk pemesanan jangka panjang atau penyesuaian harga yang dapat menarik pelanggan.
- Kembangkan pendekatan personalisasi untuk meminimalkan risiko pembatalan dari pelanggan yang memiliki riwayat pembatalan sebelumnya.
- Pertimbangkan untuk menyusun paket atau penawaran khusus untuk mengundang pelanggan untuk menginap lebih lama.



https://github.com/hubble99/Predicting-Hotel-Booking-Cancellation

ramadhoninasri09@gmail.com