LAPORAN UJIAN AKHIR SEMESTER
TI4141 – ANALITIKA DATA

Pembentukan Model Prediksi
'*Hotel Booking Cancelation*'
Beserta Usulan Strategi Bisnis

Disusun oleh:

William Wibowo Ciptono – 13418087

PROGRAM STUDI TEKNIK INDUSTRI

INSTITUT TEKNOLOGI BANDUNG

2021

PERNYATAAN KEJUJURAN PENGERJAAN UAS

"*Saya mengerjakan UAS ini secara individu, tanpa menerima bantuan dari mahasiswa TI4141 ataupun individu lainnya. Saya memahami bahwa melakukan tindak plagiarisme dan kerja sama dalam pengerjaan UAS ini adalah terlarang, dan jika saya terbukti melakukan upaya plagiarisme dan kerja sama, saya bersedia menerima sanksi sesuai dengan kebijakan ITB*."

Semarang, 29 Mei 2021

(William Wibowo Ciptono)

KATA PENGANTAR

Tulisan ini dibuat dalam rangka memenuhi ujian akhir semester dari mata kuliah TI4141 – Analitika Data. Selaku penulis, saya sangat menganjurkan untuk:

- Membaca laporan ini sambil membuka menu navigasi atau *bookmark* untuk memudahkan proses membaca.
- Mengecek ***notebook*** yang saya lampirkan karena **TIDAK SEMUA** hasil pengamatan dilampirkan pada laporan ini.

Selamat membaca,

Terima kasih

(Penulis)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR FIGUR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Sektor pariwisata kian mengalami perkembangan yang pesat seiring dengan majunya globalisasi dan terbangunnya jaringan infrastruktur seperti internet yang menjangkau seluruh kalangan masyarakat di seluruh belahan dunia. Penyebaran informasi melalui internet yang menjadi ciri khas dari era globalisasi juga turut mendukung segala usaha sosialisasi dan *word-of-mouth-marketing* mengenai tempat-tempat wisata serta pengalaman-pengalaman turis yang menarik, sedemikian rupa sehingga membawa masyarakat global untuk semakin tertarik untuk melakukan *travelling*.

Akan tetapi, dibalik semua peluang menjanjikan yang ada, terdapat masalah yang cukup signifikan dalam industri perhotelan, yaitu "pembatalan mendadak". Bahkan, data historis menunjukkan bahwa terdapat sekitar 16,4 *cancellations* per hari per satu hotel pada rentang waktu Januari hingga Juli tahun 2016 (Freed, 2016). Permasalahan tersebut sangatlah signifikan dan penting mengingat bahwa angka ini diekspektasi untuk terus bertambah dengan semakin maraknya budaya *cancel* and *re-book* reservasi hotel. Singkatnya, konsumen tidak ingin menghabiskan lebih banyak uang apabila mereka memiliki kesempatan untuk mengurangi pengeluaran mereka.

Dengan mengidentifikasi lebih dini konsumen mana yang berpeluang besar untuk melakukan pembatalan pesanan dan konsumen mana yang berpeluang kecil untuk melakukan pembatalan, dapat dilakukan pengambilan tindakan secara lebih lanjut. Pengambilan tindakan bisa dilakukan berupa pemberian insentif untuk konsumen yang memiliki peluang besar untuk melakukan pembatalan pesanan, khususnya untuk pesanan yang berada di luar *busy day*. Dengan demikian, alokasi insentif menjadi lebih terfokus dan terpersonalisasi untuk setiap konsumen dan memiliki dampak yang lebih baik secara keuangan bagi perusahaan penyedia *hospitality services* terkait.

Keadaan pandemi seperti yang sedang terjadi ketika percobaan ini dibuat (awal tahun 2021), menjadi salah satu faktor yang semakin memperlihatkan pentingnya untuk memperhatikan lebih lanjut mengenai tingkat pembatalan pesanan hotel. Hal tersebut sehubungan dengan hotel dan penyedia *hospitality services* sejenis harus semakin waspada menghadapi *growth rebound* yang akan terjadi mulai pada tahun 2022 hingga 2023 (Krishnan, Mann, Seitzman, & Wittkamp, 2020). Apabila perusahaan tidak dapat memprediksikan kemungkinan pembatalan hotel dengan baik, maka besar kemungkinan terjadi penyempitan *revenue* dan pembengkakan *opportunity cost*.

1.2. Stakeholder

Berdasarkan latar belakang permasalahan yang telah dipaparkan sebelumnya, dapat ditentukan aktor-aktor yang dianggap sebagai pemangku kepentingan dalam permasalahan terkait. Tabel 1 merupakan tabel yang menjelaskan aktor-aktor beserta kepentingannya terhadap permasalahan terkait.

Tabel 1 aktor-aktor yang berperan sebagai stakeholder

Stakeholder	**Penjelasan**
Problem owner	**Pemilik usaha perhotelan** berperan menjadi pemilik permasalahan. Kesuksesan ataupun kegagalan penanggulangan masalah akan berdampak secara langsung kepada pemilik usaha perhotelan.
Problem users	**Manajer perhotelan** berperan menjadi pengguna yang menghadapi permasalahan. Manajer perhotelan yang bertanggung jawab untuk mengatur kebijakan-kebijakan dan pengambilan keputusan yang mengarahkan perusahaan ke arah pengembangan yang lebih.
Problem customer	***Customer* perhotelan** berperan menjadi pengguna yang mengalami permasalahan. *Customer* menjadi pengguna layanan pemesanan hotel sekaligus menjadi faktor penentu apakah kamar hotel yang dipesan nantinya akan dibatalkan atau tidak.
Problem Analyst	**Penulis laporan penelitian** berperan sebagai *problem analyst* yang melakukan percobaan dengan melakukan pembuatan model prediksi pembatalan pesanan kamar hotel.

1.3. Rumusan Masalah Umum

Berikut ini merupakan rumusan permasalahan sehubungan dengan kasus yang telah dipaparkan sebelumnya:

1. Bagaimana proses perencanaan dan pelaksanaan akuisisi fitur-fitur yang berpeluang mempengaruhi terjadinya pembatalan pesanan hotel?
2. Bagaimana cara memprediksi pembatalan pesanan hotel menggunakan fitur-fitur yang ada?
3. Apa rekomendasi tindakan lanjutan yang dapat dilakukan oleh industri perhotelan khususnya dalam menangani permasalahan pembatalan pesanan?

1.4. Tujuan

Berdasarkan kondisi latar belakang dan ketersediaan data yang ada, dapat ditentukan tujuan dari penelitian ini adalah sebagai berikut:

1. Membangun model prediksi yang mampu mengklasifikasikan pesanan yang berpeluang untuk dibatalkan dan pesanan yang kurang berpeluang untuk dibatalkan.
2. Memprediksi faktor-faktor penyebab terjadinya pembatalan pemesanan kamar beserta merancang solusi perbaikan berdasarkan informasi yang diperoleh sebelumnya.

1.5. Identifikasi Sub Permasalahan

Setelah berhasil memetakan semua aspek-aspek permasalahan serta tujuan dari penelitian, selanjutnya dilakukan identifikasi mengenai detail-detail proses analitika data yang diperlukan untuk melaksanakan setiap poin permasalahan. Secara umum, permasalahan dibagi menjadi 3 (tiga) poin besar, yaitu: proses perencanaan dan pelaksanaan

akuisisi data, proses pembangunan model, dan diakhiri dengan perumusan rekomendasi tindak lanjut yang perlu dilakukan oleh pihak manajemen.

Pada tahap **proses perencanaan dan pelaksanaan akuisisi data**, terdapat 2 (dua) sub permasalahan yang harus diperhatikan, yaitu mengenai penentuan fitur-fitur apa saja yang diperlukan untuk memecahkan permasalahan terkait dan tentunya bagaimana cara memperoleh atau mengakuisisi fitur-fitur tersebut. Pada kasus pembangunan model prediksi pembatalan pesanan kamar hotel ini, khususnya pada tahapan penentuan fitur-fitur yang akan digunakan, dilakukan dengan metode studi literatur. Secara garis besar, dalam memandu proses penentuan fitur pada penelitian ini, digunakan sebuah artikel yang terdapat pada jurnal "Data in Brief" (Antonio, Almeida, & Nunes, 2019). Detail mengenai artikel terkait dan tata cara mengakuisisi data akan dijelaskan secara lebih detail pada tahap pengolahan data.

Setelah berhasil menentukan fitur-fitur penting apa saja yang diperlukan untuk proses pemodelan dan memperoleh data yang sehubungan dengan fitur-fitur terkait, proses selanjutnya adalah membangun model. Pada tahapan **pembangunan model** terdapat 3 (tiga) sub tahapan penting yang perlu diperhatikan, yaitu: proses identifikasi karakteristik data atau *data exploration*, proses pra persiapan atau *pre-processing* data, dan ditutup dengan pembuatan model itu sendiri.

Pada tahapan identifikasi karakteristik data, dilakukan teknik-teknik analisis data yang sehubungan dengan *data exploration* dan *data visualization*. Proses ini diperlukan untuk memahami bagaimana data tersebut berperilaku dan nantinya hasil pengamatan tersebut akan dijadikan landasan untuk melakukan proses pemodelan. Selain menjadi landasan untuk proses pemodelan, tahapan identifikasi karakteristik data juga berguna untuk membantu menerjemahkan hasil pemodelan sedemikian rupa sehingga dapat dihasilkan rekomendasi bagi pihak *stakeholder* permasalahan.

Pada tahapan pra preparasi data, dilakukan penanganan-penanganan yang dibutuhkan oleh data terkait sedemikian rupa sehingga siap untuk dimodelkan. Pada kasus kali ini, akan terdapat 5 (lima) aspek penting yang perlu dipertimbangkan pada proses *pre-processing* data, yaitu: penanganan *missing value*, penanganan *outlier*, pengubahan *datatype*, penanganan *imbalance data*, dan diakhiri dengan *feature engineering* & *data transformation*. Proses-proses tersebut dibutuhkan supaya data mentah yang tadinya telah berhasil diakuisisi dan diidentifikasi lebih lanjut, bisa dimodelkan dalam model-model matematis pada tahap berikutnya.

Pada tahapan pemodelan data, akan dilakukan beberapa tahapan penting yang sekiranya diperlukan untuk membangun model dengan akurasi yang baik. Pada tahapan ini juga akan ditentukan mengenai model-model apa saja yang sekiranya akan dicoba untuk memodelkan data yang tersedia. Pada kasus kali ini, terdapat 4 (empat) buah model matematis yang akan dicoba untuk digunakan, yaitu: *logistic regression*, *random forest*, *XGBoost*, dan *CatBoost*. Proses nanti akan dimulai dengan melakukan *splitting* supaya data dapat divalidasi, kemudian berlanjut ke tahapan *testing model* untuk mengecek seberapa baik performansi dari masing-masing model yang potensial. Proses akan berlanjut ke *feature selection* untuk mereduksi tingkat kompleksitas dari model. Kemudian dilanjutkan dengan *hyperparameter tuning* untuk mengoptimalkan model-model terkait, hingga akhirnya proses

pemodelan ditutup dengan pembangunan model gabungan atau *model stacking*. Proses *stacking* diberlakukan untuk mencoba menghasilkan akurasi yang lebih dengan menggabungkan beberapa model sekaligus.

Setelah berhasil membangun beberapa model yang memiliki tingkat akurasi dan kompleksitas yang bervariasi, selanjutnya akan dilakukan proses perumusan rekomendasi tindak lanjut. Pada proses perumusan rekomendasi tindak lanjut, akan dilakukan proses perbandingan tingkat kompleksitas dan akurasi dari model tersebut. Model yang baik adalah model yang dapat menghasilkan akurasi yang tinggi dengan kompleksitas yang rendah. Berdasarkan model yang dipilih, dapat ditentukan fitur-fitur yang relevan dengan menggunakan hasil kalkulasi *feature importance score*.

Berikut ini merupakan hasil penjabaran rumusan permasalahan sehubungan dengan permasalahan yang telah dipaparkan sebelumnya:

Tabel 2 identifikasi sub permasalahan beserta metode penyelesaiannya

No	Permasalahan	Sub-Permasalahan	Metode
1	Bagaimana proses perencanaan dan pelaksanaan akuisisi fitur-fitur yang berpeluang mempengaruhi terjadinya pembatalan pesanan hotel?	Bagaimana cara mengidentifikasi fitur-fitur yang relevan terhadap permasalahan?	Studi literatur
2		Bagaimana cara memastikan ketersediaan dan mengumpulkan data terkait fitur-fitur relevan yang sudah diidentifikasi sebelumnya?	Studi literatur
3	Bagaimana cara memprediksi pembatalan pesanan hotel menggunakan fitur-fitur yang ada?	Bagaimana cara mengidentifikasi karakteristik data yang sudah diperoleh sebelumnya?	*Data exploration, Data visualization*
4		Bagaimana cara menyiapkan data terkait supaya siap digunakan pada tahap pemodelan?	*Data pre-processing*
5		Bagaimana cara menentukan model-model potensial yang dapat digunakan untuk memodelkan data tersebut?	Studi literatur
6		Bagaimana cara memodelkan data terkait menggunakan beberapa pilihan model yang tersedia?	*Data modeling, cross validation*
7		Bagaimana cara menentukan fitur-fitur yang berperan secara signifikan pada masing-masing model?	*Feature Selection, RFE (Recursive Feature Elimination), SelectKBest*
8		Bagaimana cara menyesuaikan parameter untuk masing-masing model terpilih sehingga menghasilkan hasil prediksi yang optimal dan akurat?	*hyperparameter tuning, GridSearch*

9		Bagaimana cara melakukan penggabungan beberapa model terpilih untuk menghasilkan hasil yang lebih akurat?	*Model stacking, ensamble*
10	Apa rekomendasi tindakan lanjutan yang dapat dilakukan oleh industri perhotelan khususnya dalam menangani permasalahan pembatalan pesanan?	Bagaimana cara menentukan model terbaik yang dapat digunakan oleh industri perhotelan untuk memprediksi pembatalan pesanan?	*Data modeling, Model evaluation with f1_score*
11		Bagaimana cara menentukan fitur-fitur yang berperan penting dalam proses pemodelan menggunakan model terpilih tersebut?	*Feature importance analysis*
12		Bagaimana cara menentukan rekomendasi strategi bisnis berdasarkan hasil model yang diperoleh?	Analisis hasil pemodelan

BAB II
DASAR TEORI

2.1. Logistic Regression Model

Logistic regression atau Logit merupakan salah satu model regresi yang paling umum digunakan untuk memodelkan data. Secara umum, Logit merupakan pengembangan dari model *linear regression* atau Linreg yang menggunakan garis linear untuk memisahkan kategori data (Scikit Learn, 2021). Berbeda dengan Linreg, Logit merupakan model yang menghasilkan garis logistik yang memisahkan kategori data menggunakan garis logistik yang mempertimbangkan nominal fungsi **sigmoid**. Persamaan 1 fungsi sigmoid merupakan persamaan matematis untuk fungsi sigmoid.

$$logit(x, y) = \frac{1}{1 + e^{-(ax+by+c)}}$$

Persamaan 1 fungsi sigmoid

Fungsi sigmoid akan memperhitungkan jarak dari suatu poin data terhadap garis logistik yang telah dihasilkan sebelumnya. Berdasarkan jarak tersebut, dapat dihitung fungsi sigmoid yang mana menghasilkan nilai yang berada pada rentang 0 (nol) sampai 1 (satu). Semakin kecil nilai sigmoid dari suatu poin data, maka bisa diasumsikan bahwa model akan semakin yakin bahwa poin data tersebut akan bernilai 0 (nol) dan begitu pula sebaliknya.

Dalam menentukan garis yang telah dibangun oleh model Logit sudah baik atau belum, dilakukan perhitungan *likelihood*. Semakin tinggi nilai ***sumproduct*** dari likelihood seluruh poin data, maka semakin baik pula garis logistik tersebut dalam memisahkan kategori data. Persamaan 2 merupakan persamaan matematis untuk fungsi ***likelihood***.

$$likelihood = \begin{cases} p & Ketika\ nilai\ aktual = 0 \\ 1 - p & Ketika\ nilai\ aktual = 1 \end{cases}$$

Persamaan 2 fungsi likelihood

Logit sangat umum dijadikan sebagai salah satu model acuan utama dalam *data science* karena Logit memiliki berbagai kelebihan dan keuntungan. Salah satu kelebihan dari Logit adalah kebutuhan *parameter tuning* yang sangat rendah bahkan nyaris tidak ada. Berbeda kasus dengan model lain seperti *decision tree* yang apabila tidak melewati proses *pruning* dan *tuning* lainnya, model *decision tree* akan menjadi sangat rentan mengalami *overfit*.

Pada penelitian ini, khususnya dalam memodelkan data menggunakan model *logistic regression*, digunakan bantuan *science kit learn* (sklearn) *package* yang tersedia untuk bahasa pemrograman *python*, tepatnya menggunakan *sub-package 'linear_model'*. Detail mengenai *syntax* dapat diakses pada dokumentasi *package* terkait yang tersedia secara umum. Detail penerapan langsung dari *package* ini, khususnya dalam proses pemodelan data untuk memprediksi pembatalan pesanan hotel, akan dipaparkan lebih lanjut pada bab berikutnya.

2.2. Random Forest Classifier Model

Random Forest Classifier Model atau RFC, merupakan salah satu pengembangan dari model *decision tree*. Secara sederhana, RFC merupakan gabungan (*ensemble*) dari beberapa *decision tree* (Scikit Learn, 2021). Pada RFC, digunakan teknik ***bagging*** atau ***bootstrap aggregation*** (Liaw & Wiener, 2001). Pada dasarnya teknik *bootstrapping* merupakan teknik memimik data sampel secara random yang dilakukan dengan mengambil sampel data secara acak sebanyak ukuran data sesungguhnya dengan kemungkinan duplikasi data ataupun penghapusan data. Teknik *bagging* dilakukan dalam rangka menurunkan variansi model yang digunakan (singkat kata, mencuci variansi model utama dengan menggunakan data yang bervariasi untuk setiap *decision tree*, sehingga dapat didapatkan esensi dari data).

Selain melakukan *bootstrapping* pada masing-masing *decision tree* yang terdapat pada model, *random forest* juga mengarahkan untuk melakukan **dekorelasi** masing-masing *decision tree* supaya menghasilkan model yang lebih bervariasi lagi. Teknik dekorelasi dilakukan dengan membatasi fitur yang dapat dipilih secara acak pada setiap *splitting* untuk masing-masing *decision tree*. Dengan adanya pembatasan fitur secara acak, maka didapatkan model *decision tree* yang semakin bervariasi yang mana apabila digabungkan dengan *decision tree* lainnya akan menghasilkan model *random forest* yang lebih *robust* (singkat kata, mencuci variansi model utama dengan meningkatkan variasi model untuk setiap *decision tree*, sehingga didapatkan esensi dari data).

Pada penelitian ini, khususnya dalam memodelkan data menggunakan model *random forest*, digunakan bantuan *science kit learn* (sklearn) *package* yang tersedia untuk bahasa pemrograman *python*, tepatnya menggunakan *sub-package 'ensamble'*. Detail mengenai *syntax* dapat diakses pada dokumentasi *package* terkait yang tersedia secara umum. Detail penerapan langsung dari *package* ini, khususnya dalam proses pemodelan data untuk memprediksi pembatalan pesanan hotel, akan dipaparkan lebih lanjut pada bab berikutnya.

2.3. XGB Classifier Model

XGBoost merupakan salah satu model berbasis *gradient boosting* yang dikembangkan secara khusus untuk menangani permasalahan klasifikasi maupun regresi secara efisien, fleksibel, dan *portable* (readthedocs, 2021). *Gradient boosting* merupakan salah satu bentuk pengembangan dari *random forest*. Secara konsep, *gradient boosting* menggunakan teknik yang mirip dengan *random forest*, yaitu dengan menggabungkan beberapa model menjadi satu meta model menggunakan teknik *ensamble*. Perbedaan dari *gradient boosting* terletak pada kemampuan untuk melakukan optimasi pada *cost function* melalui diferensiasi *loss function*.

Pada penelitian ini, khususnya dalam memodelkan data menggunakan model *logistic regression*, digunakan bantuan *xgboost package* yang tersedia untuk bahasa pemrograman *python*. Detail mengenai *syntax* dapat diakses pada dokumentasi *package* terkait yang tersedia secara umum. Detail penerapan langsung dari *package* ini, khususnya dalam proses pemodelan data untuk memprediksi pembatalan pesanan hotel, akan dipaparkan lebih lanjut pada bab berikutnya.

2.4. CatBoost Classifier Model

CatBoost merupakan salah satu model yang digunakan untuk menangani permasalahan klasifikasi (catboost, 2021). Berbeda dengan XGBoost, CatBoost menggunakan metode *sequential learning boosting* yang mana merupakan pengembangan lebih lanjut dari *gradient boosting* (Dorogush, Ershov, & Yandex, 2018). Perbedaan terletak pada *weak model* yang digunakan untuk membangun CatBoost. CatBoost menggunakan gabungan dari berbagai *decision tree* (mirip seperti *random forest*) akan tetapi CatBoost tetap mempertahankan ciri khas model *gradient boosting* yang mampu melakukan optimasi pada *cost function* melalui diferensiasi *loss function*.

Pada penelitian ini, khususnya dalam memodelkan data menggunakan model *logistic regression*, digunakan bantuan *catboost package* yang tersedia untuk bahasa pemrograman *python*. Detail mengenai *syntax* dapat diakses pada dokumentasi *package* terkait yang tersedia secara umum. Detail penerapan langsung dari *package* ini, khususnya dalam proses pemodelan data untuk memprediksi pembatalan pesanan hotel, akan dipaparkan lebih lanjut pada bab berikutnya.

2.5. Feature Selection

Feature selection atau pemilihan fitur, merupakan salah satu proses mengeliminasi baik secara otomatis ataupun manual, fitur-fitur yang bersifat *noise* dan memilih fitur-fitur yang paling berkontribusi terhadap *prediction variable* (Shaikh, 2018). Membiarkan fitur-fitur yang bersifat *noise* akan menyebabkan akurasi model menjadi menurun. Hal tersebut terjadi karena model juga turut memperhitungkan data-data yang kurang relevan dalam proses pemodelan.

Seperti yang sudah dijelaskan di awal, *feature selection* bisa dilakukan secara otomatis ataupun manual. Pada kasus pemilihan fitur secara manual, diperlukan kemampuan analisis dan keilmuan dari para ahli mengenai fitur terkait. Sedangkan untuk pemilihan secara otomatis, dapat digunakan beberapa teknik baik itu berupa teknik yang bersifat statistik maupun teknik yang bersifat rekursif. Selanjutnya akan dibahas mengenai teknik-teknik pemilihan fitur secara otomatis yang digunakan dalam penelitian ini.

2.5.1. RFECV

Salah satu cara melakukan *feature selection* secara otomatis adalah dengan menggunakan teknik RFE atau *recursive feature elimination*. Secara sederhana, teknik ini merupakan teknik bersifat *brute-force*, model akan menggunakan n fitur yang mana akan dicoba satu per satu untuk terus meningkatkan jumlah n mulai dari 1 (satu) hingga jumlah akhir dari fitur. Kemudian hasil dari masing-masing percobaan akan menghasilkan akurasi model yang berbeda-beda. Dengan bertambahnya n, seharusnya nilai akurasi juga akan terus bertambah atau menjadi semakin baik. Akan tetapi, akan ada pada waktunya dimana penambahan n yang berkelanjutan justru menurunkan nilai akurasi model. Di saat seperti itu lah, fitur-fitur tersebut diasumsikan bersifat *noise* dan sebaiknya tidak digunakan dalam pemodelan.

Pada kasus RFECV atau *recursive feature elimination cross validation,* proses validasi untuk masing-masing fitur dilakukan dengan menggunakan teknik pengecekan akurasi *cross validation.* Secara singkat, *cross validation* adalah proses memecah *dataset* menjadi

sebanyak k bagian (biasa juga dibilang k lipat atau *k fold*), kemudian dilakukan duplikasi sebanyak k untuk data terkait dan dilakukan proses *cross validate* atau pengecekan satu sama lain (1 lipatan menjadi data tes dan k-1 data menjadi data latihan, diulang untuk setiap lipatan).

Pada kasus penelitian ini, dipilih untuk menggunakan RFECV daripada RFE yang divalidasi secara manual menggunakan *fit* dan *score* karena keterbatasan jumlah data. Secara umum, terdapat berbagai parameter yang dapat dikendalikan pada RFECV seperti 'step' yang mengendalikan pertambahan variabel n, kemudian ada juga parameter k yang mengendalikan jumlah lipatan, 'random_state' yang mengatur randomisasi data, dan lain-lain.

Pada penelitian ini, khususnya dalam melakukan *feature selection* dengan teknik RFECV, digunakan bantuan science kit learn (sklearn) package yang tersedia untuk bahasa pemrograman python, tepatnya menggunakan sub-package 'feature_selection'. Detail mengenai *syntax* dapat diakses pada dokumentasi package terkait yang tersedia secara umum. Detail penerapan langsung dari package ini, khususnya dalam proses pemodelan data untuk memprediksi pembatalan pesanan hotel, akan dipaparkan lebih lanjut pada bab berikutnya.

2.5.2. SelectKBest

Selain menggunakan teknik RFECV, juga dilakukan proses seleksi fitur menggunakan teknik statistika dengan bantuan fungsi SelectKBest. Secara sederhana, SelectKBest akan menggunakan uji statistika seperti *chi-squared* ataupun *f testing* untuk menentukan fitur mana yang berkorelasi aktif dengan variabel target. Setelah menentukan korelasi untuk masing-masing fitur, skor atau nilai korelasi tersebut akan diurutkan dari yang tertinggi ke yang terendah. Kemudian dipilih sejumlah n fitur dengan tingkat korelasi tertinggi.

Pada penelitian ini, khususnya dalam melakukan *feature selection* dengan teknik SelectKBest, digunakan bantuan science kit learn (sklearn) package yang tersedia untuk bahasa pemrograman python, tepatnya menggunakan sub-package 'feature_selection'. Detail mengenai *syntax* dapat diakses pada dokumentasi package terkait yang tersedia secara umum. Detail penerapan langsung dari package ini, khususnya dalam proses pemodelan data untuk memprediksi pembatalan pesanan hotel, akan dipaparkan lebih lanjut pada bab berikutnya.

2.6. Hyperparameter Tuning

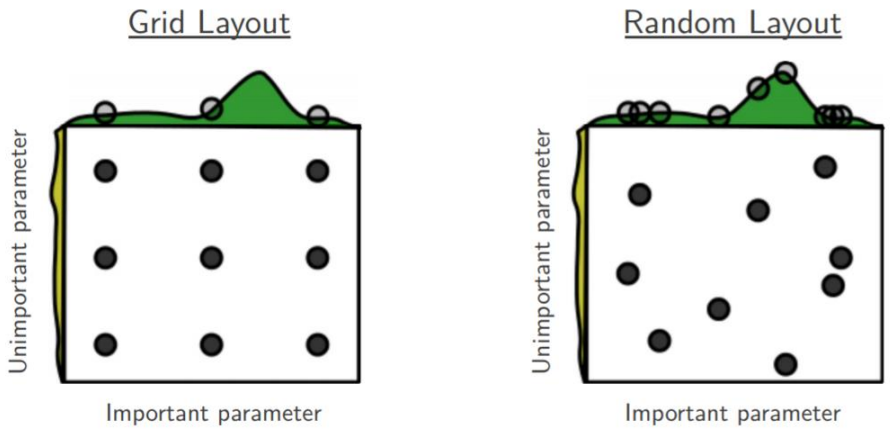
Parameter adalah variabel konfigurasi yang bersifat internal di dalam model yang nilainya dapat diperoleh dari proses *fitting* data. Berbeda dengan parameter yang umum dikenal di dalam model, terdapat juga istilah *hyperparameter*. *Hyperparameter* merupakan parameter konfigurasi eksternal yang nilainya tidak dapat diestimasi atau diperoleh melalui proses *fitting* data (Paul, 2018).

Dalam penerapannya, beberapa model yang disediakan pada *package-package* pemrograman sudah memiliki nilai *default* dari *hyperparameter*. Akan tetapi, karena *hyperparameter* tersebut bersifat eksternal dan tidak dapat diperoleh melalui proses *fitting data*, maka perlu dilakukan proses *hyperparameter tuning* untuk menyesuaikan

hyperparameter terkait sedemikian rupa sehingga dapat menghasilkan model dengan hasil prediksi yang optimal.

Secara umum, *hyperparameter* dapat diatur atau disesuaikan secara manual maupun secara otomatis. Pada kasus penyesuaian secara manual, diperlukan *labor force* yang cukup intensif dan diperlukan pengetahuan khusus mengenai model, data, dan permasalahan terkait. Pada keadaan aktual, penyesuaian secara manual sangat jarang dilakukan karena sering kali tidak menghasilkan hasil yang baik dan optimal dikarenakan banyaknya variabilitas di dalam penentuan *hyperparameter*.

Pada penyesuaian secara otomatis, terdapat 2 (dua) teknik yang umum digunakan untuk melakukan *hyperparameter tuning*. Kedua teknik tersebut merupakan teknik bernama *Grid Search* dan *Random Search*. Gambar 1 menunjukan perbedaan antara kedua teknik. Secara umum, sangat disarankan untuk menggunakan *Random Search* pada kasus dataset berukuran besar. Pada kasus pembangunan model prediksi pembatalan pesanan hotel yang memiliki dataset cenderung kecil, digunakan teknik *Grid Search Cross Validation* atau GridSearchCV.



Gambar 1 perbandingan antara Grid Search dengan Random Search

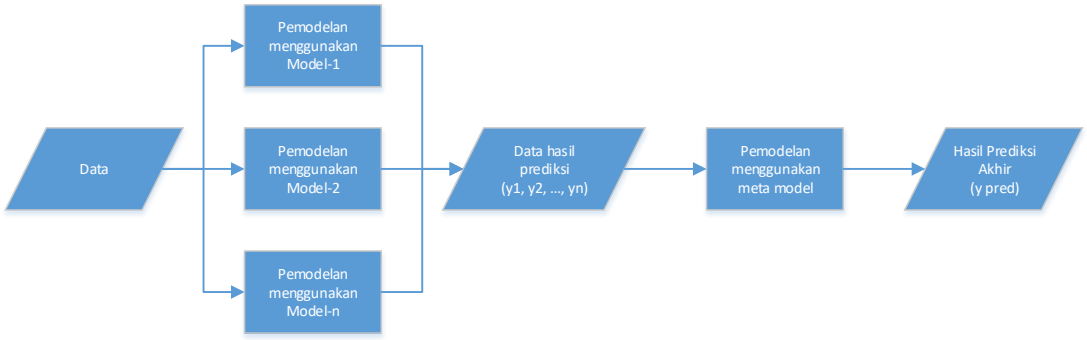
Pada penelitian ini, khususnya dalam melakukan *Hyperparameter tuning* dengan teknik *Grid Search Cross Validation*, digunakan bantuan science kit learn (sklearn) package yang tersedia untuk bahasa pemrograman python, tepatnya menggunakan sub-package 'model_selection'. Detail mengenai *syntax* dapat diakses pada dokumentasi package terkait yang tersedia secara umum. Detail penerapan langsung dari package ini, khususnya dalam proses pemodelan data untuk memprediksi pembatalan pesanan hotel, akan dipaparkan lebih lanjut pada bab berikutnya.

2.7. Model Ensemble / Stacking

Model stacking adalah teknik yang digunakan untuk mengkombinasikan hasil prediksi dari berbagai model untuk menghasilkan suatu meta model yang mampu memiliki akurasi prediksi yang lebih baik (Himmetoglu, 2017). Proses menggabungkan hasil prediksi dari beberapa model berlandaskan pada konsep saling melengkapi, dimana masing-masing model dapat saling melengkapi dalam hal memprediksi poin-poin data yang mungkin sulit

untuk dikategorisasi bagi sebagian model, tapi sebenarnya cenderung lebih mudah dikategorisasikan oleh model lain.

Pada dasarnya, *model stacking* bermula dari proses pelatihan banyak model yang akan digabungkan. Setelah dilatih, maka dapat dilakukan proses prediksi untuk satu data latihan tertentu. Hasil prediksi dari masing-masing model akan disatukan kembali menjadi satu data kumpulan hasil prediksi. Kumpulan hasil prediksi inilah yang nantinya akan dimodelkan oleh sebuah meta model hingga nantinya menghasilkan hasil prediksi akhir. Gambar 2 merupakan visualisasi sederhana dari konsep dasar *model stacking*.

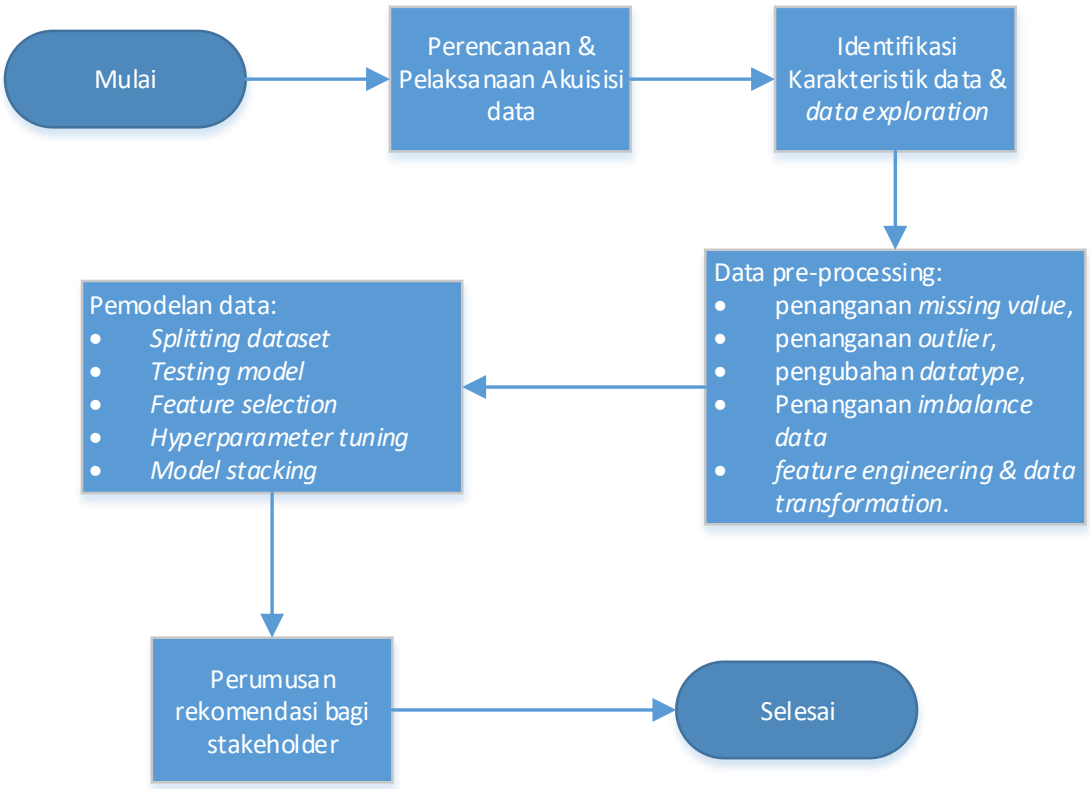


Gambar 2 visualisasi konsep dasar model stacking

BAB III
METODOLOGI PENELITIAN

Penelitian pembangunan model prediksi pembatalan pesanan kamar hotel, dilakukan dalam beberapa tahapan. Secara umum, proses penelitian dimulai dari identifikasi data-data apa yang diperlukan beserta asal muasal dari data terkait. Setelahnya dilakukan proses akuisisi data sehingga data bisa digunakan dalam proses pemodelan. Setelah melakukan proses akuisisi, proses dilanjutkan dengan tahapan identifikasi karakteristik data atau biasa disebut juga dengan *data exploration*.

Setelah memahami karakteristik dari data, maka penelitian dapat dilanjutkan ke tahapan *pre-processing*. Pada tahapan *data pre-processing*, terdapat 5 (lima) sub tahapan yaitu: penanganan *missing value*, penanganan *outlier*, pengubahan *datatype*, penanganan *imbalance data* dan *feature engineering*. Setelah menyelesaikan tahapan *pre-processing*, proses pembangunan model dapat dilanjutkan ke tahapan pemodelan data. Terdapat 5 (lima) sub tahapan pada tahap pemodelan data yang terdiri atas: *splitting dataset, testing model, feature selection, hyperparameter tuning* dan *model stacking*. Setelah berhasil melakukan proses pemodelan data, maka dapat ditarik kesimpulan berupa rumusan rekomendasi bagi stakeholder yang sekaligus menutup alur proses penelitian ini. Figur 1 merupakan visualisasi urutan proses penelitian ini.



Figur 1 diagram urutan proses pelaksanaan penelitian pembangunan model prediksi pembatalan pesanan hotel

BAB IV
PENGOLAHAN DATA

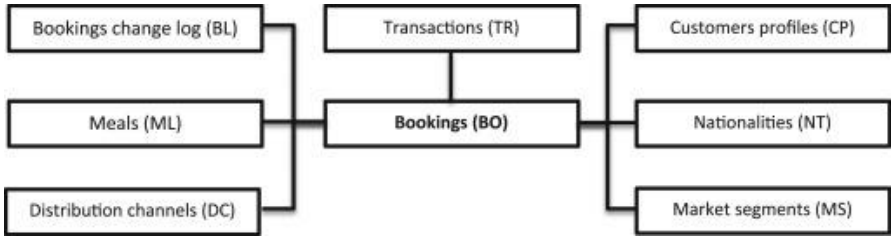
4.1. Proses Perencanaan & Pelaksanaan Akuisisi Data

Sebelum melakukan pemodelan untuk pemecahan masalah, perlu dilakukan analisis identifikasi kebutuhan dan ketersediaan data terkait yang dibutuhkan untuk proses pemodelan. Pada kasus kali ini, digunakan pendekatan studi literatur untuk mencari fitur dan memastikan ketersediaan data terkait yang akan digunakan untuk proses pemodelan.

Setelah dilakukan pencarian lebih lanjut untuk data pemesanan hotel, ditemukan sebuah dataset mengenai pemesanan hotel di Portugal. Data ini merupakan dataset Hotel Booking Demand yang diambil dari situs Kaggle. Kaggle merupakan platform penyedia layanan berbagi *datasets*, eksplorasi, dan pemodelan. Gambar 3 merupakan logo dari platform Kaggle. Berikut merupakan tautan dari laman Kaggle yang membagikan informasi mengenai dataset Hotel Booking Demand (https://www.kaggle.com/jessemostipak/hotel-booking-demand).

Gambar 3 logo platform berbagi datasets, eksplorasi, dan pemodelan, Kaggle

Data yang tersedia pada situs Kaggle tersebut merupakan data yang cukup kredibel karena diambil dari salah satu artikel bernama "Hotel Booking Demand Dataset" pada halaman 41-49 jurnal bernama "Data In Brief" Volume 22 yang terbit pada Februari 2019 kemarin. Dataset yang digunakan merupakan **gabungan** dari **2 (dua) dataset** hotel demand H1 (Resort Hotel) dan H2 (City Hotel) yang masing-masing berlokasi di Portugal (Antonio, Almeida, & Nunes, 2019). Data yang terdapat pada masing-masing *dataset* hotel merupakan gabungan dari berbagai data yang berasal dari berbagai *database* hotel terkait. Gambar 4 merupakan gambar visualisasi basis data yang digunakan pada proses pengambilan dataset.



Gambar 4 visualisasi basis data hotel yang digunakan pada pengambilan dataset

Secara umum, terdapat 40060 row data pengamatan yang diambil dari H1 dan 79330 row data pengamatan yang diambil dari H2 pada rentang waktu 1 Juli 2015 s/d 31 Agustus 2017. Secara total, dataset yang digunakan mengandung 31 (tiga puluh satu) kolom feature dengan 1 (satu) target akhir berupa kolom "is_canceled" yang menandakan apakah konsumen terkait melakukan pembatalan pesanan hotel atau tidak. Perihal detail dari masing-masing feature yang terdapat pada dataset terkait akan dipaparkan pada tabel di bawah ini:

Tabel 3 feature dictionary

No	Fitur	*Datatype*	Deskripsi
1	ADR	Float	*Average Daily Rate* = Jumlah transaksi dalam sehari dibagi dengan jumlah malam menginap
2	Adults	Integer	Jumlah orang dewasa
3	Agent	Categorical	ID travel agen yang melakukan pemesanan
4	Arrival Date Day of Month	Integer	Hari kedatangan (1 sampai 31)
5	Arrival Date Month	Categorical	Bulan kedatangan mulai dari Januari hingga Desember
6	Arrival Date Week Number	Integer	Minggu kedatangan (1 sampai 52)
7	Arrival Date Year	Integer	Tahun kedatangan
8	Assigned Room Type	Categorical	Tipe kamar yang didapatkan oleh *customer*. Terkadang nilai *assigned room type* berbeda dengan permintaan konsumen.
9	Babies	Integer	Jumlah bayi
10	Booking Changes	Integer	Jumlah perubahan atau revisi detail pesanan
11	Children	Integer	Jumlah anak-anak
12	Company	Categorical	ID dari perusahaan yang memesan kamar hotel atas nama perusahaan
13	Customer type	Categorical	Diasumsikan terdapat 4 (empat) jenis konsumen, yaitu: • *Contract* → pesanan kamar terikat dengan kontrak atau sejenisnya • *Group* → pesanan kamar terasosiasi dengan sebuah grup • *Transient* → pesanan kamar yang tidak terikat kontrak ataupun grup, tidak juga memiliki asosiasi antar pesanan yang bersifat transient • *Transient-party* → pesanan kamar yang tidak terikat kontrak ataupun grup, tapi memiliki asosiasi dengan pesanan yang bersifat transient lainnya
14	Days in waiting list	Integer	Lama hari pesanan berada pada *waiting list*
15	Deposit type	Categorical	Terdapat 3 (tiga) jenis deposit, yaitu:

No	Fitur	*Datatype*	Deskripsi
			• No Deposit → tidak ada deposit sama sekali • Non Refund → ada deposit yang melebihi ongkos pesanan kamar hotel • Refundable → ada deposit sebagian ongkos pesanan kamar hotel
16	Distribution channel	Categorical	Saluran distribusi pesanan hotel
17	Is canceled	Categorical	Variabel target yang akan diprediksi, jika bernilai 1 (satu) artinya pesanan dibatalkan dan begitu pula sebaliknya.
18	Is repeated guest	Categorical	Ketika pemesan memiliki akun member dan sudah pernah memesan sebelumnya
19	Lead time	Integer	Selisih jumlah hari pencatatan data pesanan ke sistem dengan tanggal kedatangan
20	Market segment	Categorical	Segmentasi pasar
21	Meal	Categorical	Terdapat 4 (empat) jenis paket makanan, yaitu: • Undefined / SC → tidak pesan • BB → *Bed & Breakfast* • HB → *Half Board* (*breakfast* & satu makan siang atau malam, biasanya makan malam) • FB → *Full Board* (*breakfast, lunch,* dan *dinner*)
22	Previous booking not canceled	Integer	Jumlah pemesanan sukses dari *customer* yang sama
23	Previous canceled	Integer	Jumlah pesanan yang pernah dibatalkan dari *customer* yang sama
24	Required Car Parking Space	Integer	Jumlah tempat parkir yang diminta oleh konsumen
25	Reservation Status	Categorical	Terdapat 3 (tiga) jenis status pesanan, yaitu: • *Canceled* • *Check-out* • *No-Show*
26	Reservation Status Date	Date	Tanggal pengubahan status terakhir
27	Reserved room type	Categorical	Jenis kamar yang dipesan oleh konsumen
28	Stay in weekend night	Integer	Jumlah malam akhir pekan (sabtu & minggu) yang dipesan
29	Stay in week night	Integer	Jumlah malah di hari kerja (senin sampai jumat) yang dipesan

No	Fitur	*Datatype*	Deskripsi
30	Total of special request	Integer	Jumlah permintaan spesial yang diajukan oleh konsumen (*twin bed, high floor,* dll.)

Data yang sudah dipaparkan sebelumnya kemudian dimasukkan ke dalam IDE untuk proses pengolahan lebih lanjut menjadi model prediksi pembatalan pesanan hotel. Proses imputasi dataset terkait dilakukan dengan cara mengunduh terlebih dahulu dataset terkait via situs *platform* Kaggle dalam format csv (*comma separated values*), kemudian dilakukan imputasi ke dalam IDE. Selain memasukkan dataset terkait ke dalam IDE, dilakukan juga proses memasukkan *package* yang diperlukan untuk proses pemodelan. Berikut merupakan *syntax* yang digunakan untuk melakukan kedua perintah imputasi yang telah dipaparkan sebelumnya dalam bahasa pemrograman *Python:*

```
# Importasi library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import seaborn as sns
import pycountry
import pycountry_convert as pc

# imputasi dataset
data = pd.read_csv("hotel_bookings.csv")
```

4.2. Proses Pembuatan Model Prediksi

Setelah menyiapkan data yang akan diolah ke dalam model prediksi pembatalan pesanan hotel, proses selanjutnya adalah pembangunan model prediksi itu sendiri. Proses pembangunan model prediksi akan dilakukan dalam beberapa tahap sesuai dengan yang telah dipaparkan pada Figur 1 diagram urutan proses pelaksanaan penelitian pembangunan model prediksi pembatalan pesanan hotel.

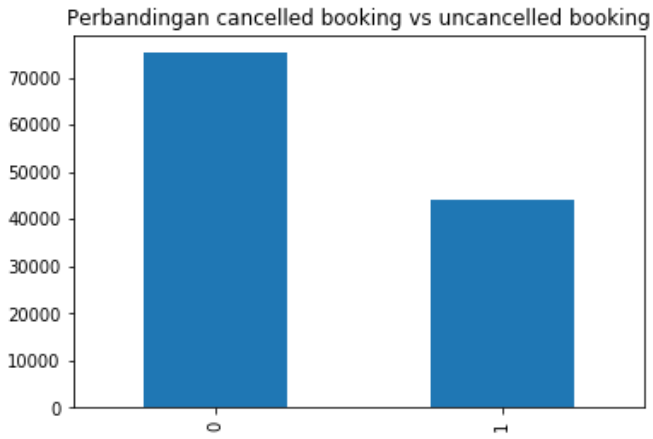
4.2.1. Identifikasi karakteristik data

Salah satu hal penting yang perlu dilakukan sebelum benar-benar melakukan proses pemodelan, adalah memahami data yang diolah itu sendiri. Proses atau tahapan ini sering kali disebut sebagai ***data exploration & business understanding***. Banyak informasi yang dapat diperoleh pada tahapan *data exploration*, mulai dari identifikasi bentuk dari dataset, identifikasi tipe data masing-masing fitur, identifikasi adanya *missing value*, identifikasi adanya *outlier*, identifikasi korelasi antara fitur terkait dengan variabel target, dan lain-lain.

Secara umum terdapat **20 (dua puluh) poin analisis** pada tahapan *data exploration* yang telah berhasil dilakukan untuk dataset terkait. Detail mengenai tahapan identifikasi karakteristik data dapat dibaca pada *notebook* (format .ipynb) yang dilampirkan di luar laporan ini. Berikut merupakan beberapa cuplikan *syntax* yang digunakan pada tahapan identifikasi karakteristik data:

```
# 1. informasi umum dataset

data.shape
```

```
data.describe()

data.info()

data.isna().sum()

data['is_canceled'].value_counts().plot(kind='bar',
title="Perbandingan cancelled booking vs uncancelled booking")
```



Figur 2 grafik perbandingan proporsi pesanan yang dibatalkan dengan pesanan yang tidak dibatalkan

Berdasarkan informasi umum yang digali, didapati beberapa hasil observasi seperti:

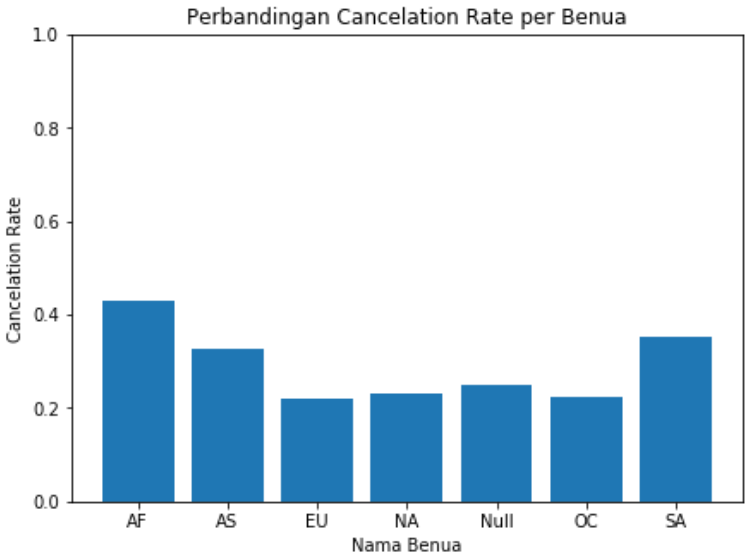
- Didapatkan bahwa terdapat beberapa *missing value*, khususnya pada kolom *children, country, agent,* dan *company*. *Missing value* pada fitur *company* dan *agent* berukuran sangat besar, pertimbangkan untuk *drop 2 feature* tersebut.
- Nilai rata-rata kolom target 'is_canceled' tidak rata di angka 0,5. Menandakan bahwa komposisi data *booking* yang berujung pada 'is_canceled' = True dan 'is_canceled' = False tidaklah rata. Perlu dipertimbangkan untuk menggunakan parameter penilaian khusus seperti *F1 Score* supaya hasil output menjadi lebih kredibel.
- Tipe data untuk *feature* 'resevation_status_date', 'arrival_date_month' masih dalam bentuk *object / string*. Kemungkinan perlu dilakukan pengubahan tipe data lebih lanjut untuk mempermudah proses pengolahan.
- Masih banyak beberapa fitur lain yang masih dalam bentuk *object*, nantinya perlu dilakukan *encoding* sebelum masuk ke tahap modelling.
- Data sedikit kurang *balance*, lebih banyak kasus *cancelation booking* daripada *booking* yang sukses.

Berdasarkan hasil pengamatan, dapat ditarik kesimpulan:

- Periksa kembali *feature* 'agent' & 'company'
- Perbaiki *missing value* 'children' & 'country' atau hapus row data bersangkutan
- Gunakan parameter penilaian F1 Score

- Ubah *datatype* 'reservation_status_date' & 'arrival_date_month' ke dalam bentuk *datetime* apabila dibutuhkan & memungkinkan
- Lakukan *encoding* untuk beberapa fitur dengan *data-type object/string*

```
# 8. Analisis pengaruh asal lokasi negara pemesan
# fungsi convert benua
def convert_benua(x):
    if x == "TMP": # khusus timor leste, karena ganti nama iso
code
        return "AS"
    if len(x) == 3:
        x=pc.country_alpha3_to_country_alpha2(x)
    #khusus untuk negara tanpa benua:
    if x in ['AQ', 'UM', 'TF']:
        return "Null"
    return pc.country_alpha2_to_continent_code(x)

# pengelompokan data berdasar benua
data_benua = data.loc[data['country']!='PRT',
['is_canceled','country']].dropna()
data_benua['benua'] =
data_benua['country'].apply(convert_benua)


# perhitungan rata-rata cancelation rate berdasar benua
data_per_benua = data_benua[['benua',
'is_canceled']].groupby('benua', as_index=False).mean()

fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(111)

ax.set_title("Perbandingan Cancelation Rate per Benua")
ax.set_ylim([0,1])
ax.set_ylabel("Cancelation Rate")
ax.set_xlabel("Nama Benua")
ax.bar(data_per_benua['benua'], data_per_benua['is_canceled'])
```



Figur 3 grafik perbandingan rata-rata pembatalan pesanan per benua

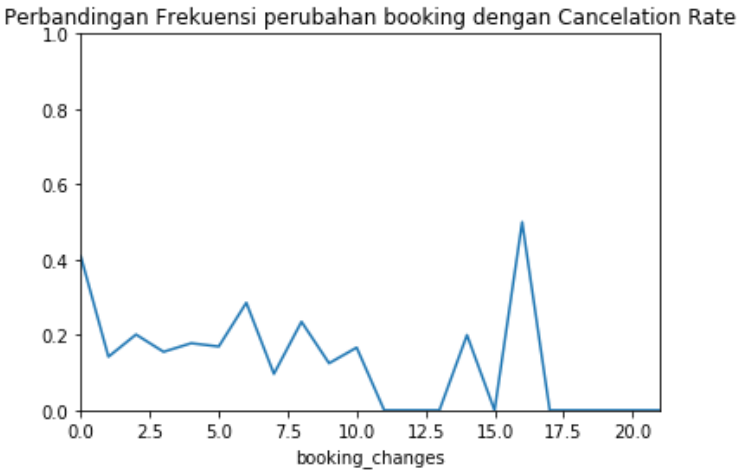
Berdasarkan informasi umum yang digali, didapati beberapa hasil observasi seperti:

- Pemesan secara garis besar terpusat pada PRT > GBR > FRA > ESP > DEU (Portugal, United Kingdom, France, Spain, Germany).
    - Portugal = Tempat pengambilan data / lokasi hotel;
    - UK & Germany = Dekat dengan Portugal & GDP relatif tinggi;
    - France & Spain = Dekat dengan Portugal
- Dari seluruh negara yang memiliki pemesanan lebih dari 1000, PRT memiliki tingkat pembatalan yang paling tinggi. Indikasi pemesanan dalam negeri memiliki peluang pembatalan yang lebih tinggi
- Secara umum, negara asal mempengaruhi tinggal pembatalan order. Terlihat dari besar peluang pembatalan yang bervariasi antar negara.
- Ketika diklasifikasikan menurut benua asal, ditemukan bahwa AF (Africa) memiliki tingkat Cancelation yang lebih besar daripada benua lainnya, dilanjut dengan SA (South America) dan AS (Asia). Diduga hal ini dipengaruhi oleh budaya pemesanan dari masing-masing kontinen terkait.

Berdasarkan hasil pengamatan, dapat ditarik kesimpulan:

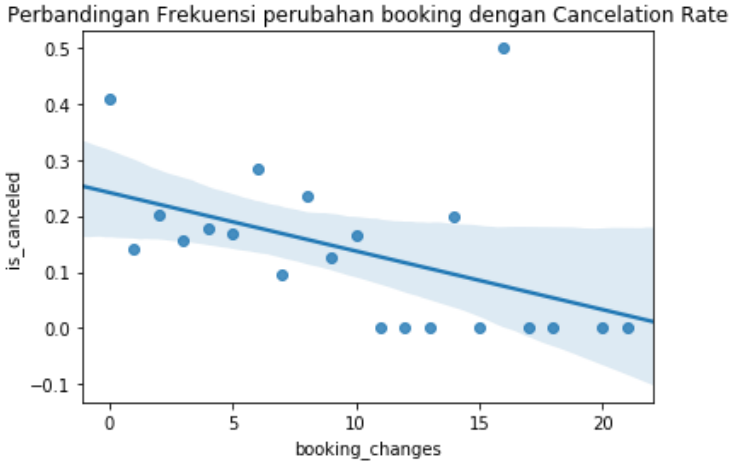
Dapat dipertimbangkan untuk mengklasifikasikan negara berdasarkan benuanya guna menurunkan kompleksitas model, tapi tetap membiarkan menyediakan 1 fitur khusus untuk negara "Portugal" selaku indikator pemesanan dalam negeri yang mana memiliki peluang cancel yang relatif tinggi.

```
# 12. Analisis pengaruh frekuensi perubahan pesanan
data.groupby('booking_changes')['is_canceled'].mean().plot(yli
m=([0,1]), title="Perbandingan Frekuensi perubahan booking
dengan Cancelation Rate")
```



Figur 4 grafik perbandingan frekuensi perubahan booking terhadap peluang pembatalan pesanan

```
data_booking_change = data.groupby('booking_changes',
as_index=False)['is_canceled'].mean()
sns.regplot(data_booking_change['booking_changes'],
data_booking_change['is_canceled']).set_title("Perbandingan
Frekuensi perubahan booking dengan Cancelation Rate")
```



Figur 5 grafik perbandingan frekuensi perubahan booking terhadap tren peluang pembatalan pesanan

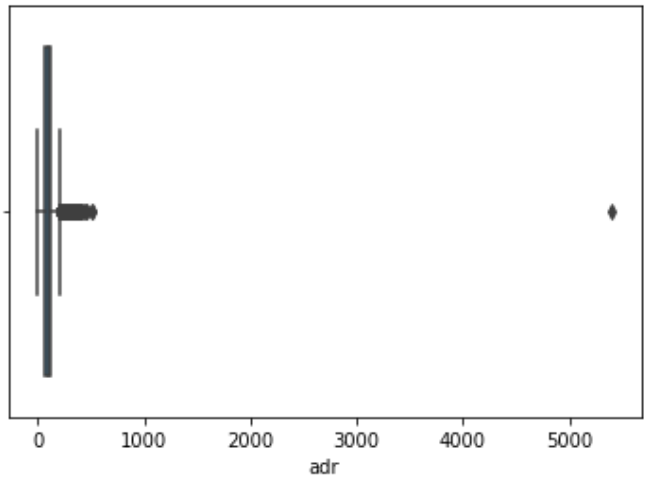
Berdasarkan informasi umum yang digali, didapati beberapa hasil observasi seperti:

- Orang yang setidaknya pernah melakukan perubahan booking sekali, memiliki peluang cancelation yang lebih rendah. Mungkin ini menandakan pola perilaku ketika orang sudah merubah booking mereka, hal tersebut menandakan bahwa mereka menaruh effort & lebih cenderung untuk mempertahankan booking tersebut untuk kedepannya.
- Secara umum, semakin banyak jumlah booking changes, semakin berkurang kemungkinan cancel booking

Berdasarkan hasil pengamatan, dapat ditarik kesimpulan:

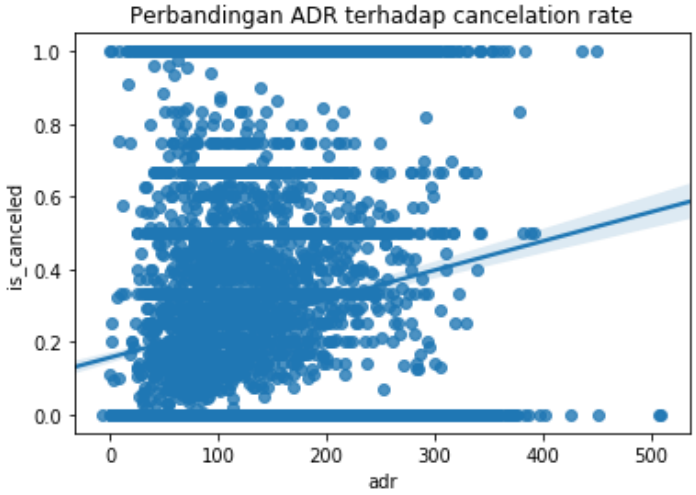
- Pertahankan fitur booking changes
- Tambahkan boolean dari "pernah merubah book" atau tidak

```
# 18. Analisis pengaruh ADR (Average Daily Rate)
sns.boxplot(data['adr'])
```



Figur 6 grafik boxplot persebaran data ADR

```
data_adr = data.loc[data['adr']< 1000].groupby(['adr'],
as_index=False)['is_canceled'].mean()
sns.regplot(data_adr['adr'],
data_adr['is_canceled']).set_title("Perbandingan ADR terhadap
cancelation rate")
```



Figur 7 grafik perbandingan ADR terhadap rasio pembatalan pesanan

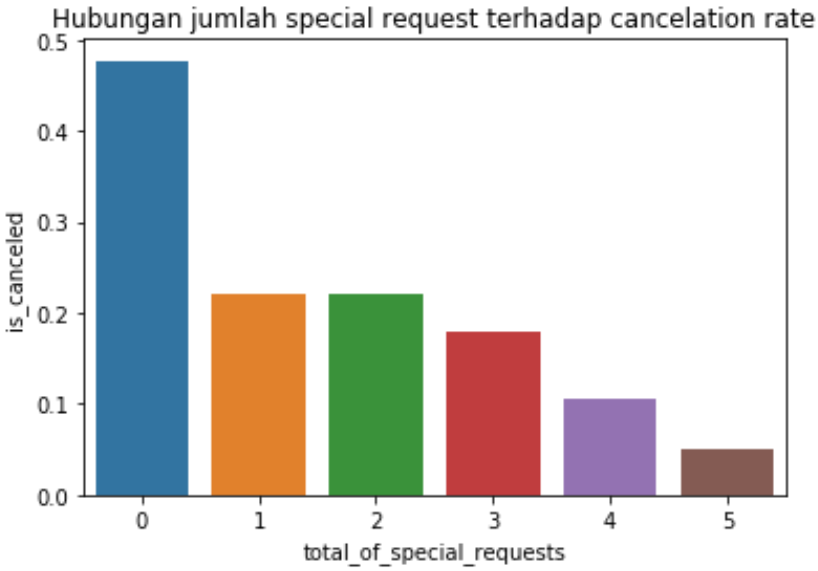
Berdasarkan informasi umum yang digali, didapati beberapa hasil observasi seperti:

- Terdapat 2 row data outlier yang berpotensi menjadi noise jika diabaikan
- Terdapat tren positif antara ADR dengan tingkat pembatalan

Berdasarkan hasil pengamatan, dapat ditarik kesimpulan:

- drop data outlier
- pertahankan fitur ADR

```
# 20. Analisis pengaruh jumlah special request
data_req = data.groupby('total_of_special_requests',
as_index=False)['is_canceled'].mean()
sns.barplot(data_req['total_of_special_requests'],
data_req['is_canceled']).set_title("Hubungan jumlah special
request terhadap cancelation rate")
```



Figur 8 grafik perbandingan tingkat pembatalan terhadap jumlah special request

Berdasarkan informasi umum yang digali, didapati beberapa hasil observasi seperti semakin banyak jumlah *special request*, semakin kecil kemungkinan pembatalan order. Berdasarkan hasil pengamatan tersebut, dapat ditarik kesimpulan untuk mempertahankan fitur jumlah *special request.*

4.2.2. *Data pre-processing*

Setelah berhasil memahami data yang dimiliki, proses selanjutnya adalah melakukan *data pre-processing*. Definisi dari *data pre-processing* adalah sebuah proses dimana data mengalami transformasi, *encoding,* ataupun proses lainnya, sedemikian rupa sehingga data menjadi pada kondisi siap dipakai dalam proses pemodelan (Pandey, 2019). Pada kasus pemodelan data pembatalan pesanan hotel, dilakukan 5 (lima) tahap *pre-processing* yang terdiri dari: penanganan *missing value*, penanganan *outlier*, penanganan *datatype*, penanganan *imbalance data* dan *feature engineering*.

4.2.2.1. Penanganan *missing value*

Kasus *missing value* merupakan kasus yang sangat umum terjadi dalam proses pengolahan data. *Missing value* dipengaruhi oleh banyak faktor seperti *system error, hardware error,* ataupun *human error*. Pada bahasa pemrograman Python, cara mengecek apabila terdapat *missing value* pada suatu dataset, dapat menggunakan *syntax* sebagai berikut:

```
# mengecek missing value
data.isna().sum()
```

Berdasarkan jenisnya, kasus *missing value* dibagi menjadi 4 (empat) jenis, yaitu: *structural missing data, missing completely at random (MCAR), missing at random (MAR),* dan *missing not at random* (Pandey, 2019). Dalam menangani kasus *missing value*, sangat penting untuk dapat mengidentifikasi tipe jenis *missing value*, khususnya pada kasus-kasus *missing not at random* sangat diperlukan penanganan dan perhatian lebih lanjut. **Apabila dibiarkan**, jenis-jenis *missing value* tertentu, khususnya yang bertipe *missing not at random*, akan **mempengaruhi dan berdampak buruk pada hasil prediksi dari model**. Beruntung, pada kasus *dataset* yang sedang diolah, tidak terdapat indikasi *missing value* berjenis *missing not at random*.

Data yang tergolong pada *structural missing data* merupakan data-data yang secara logika memang hilang karena struktur penyimpanan basis data. Pada kasus kali ini, terdapat *missing value* berjenis *structural missing data* pada fitur 'agent' sebanyak 16340 data dan 'company' sebanyak 112593 data. Hal tersebut terjadi dikarenakan ada beberapa pemesanan yang memang tidak menggunakan agen dan bukanlah juga pemesanan yang berasal dari suatu perusahaan tertentu.

Pada kasus MCAR dan MAR, *missing data* masih dapat ditoleransi tergantung seberapa besar data yang hilang. Menurut IBM, *missing value* yang berukuran kurang dari 5% data sesungguhnya masih dapat dikategorikan sebagai MCAR dan MAR (IBM, 2021). Pada kasus ini, fitur 'country' dan 'children' memiliki *missing value* dalam ukuran kecil (4 data untuk fitur 'children' dan 488 data untuk fitur 'country').

Terdapat beberapa teknik penanganan *missing value*, diantaranya: melakukan penghapusan *row data* yang memiliki *missing value*, imputasi, estimasi, dan terakhir penghapusan fitur. Pada kasus pembuatan model prediksi pembatalan pesanan hotel, digunakan teknik imputasi untuk fitur 'agent' dan 'company' yang memiliki *missing value* berjenis *structural missing data*. Sementara untuk fitur 'country' dan 'children', dilakukan penghapusan *row data* dengan mempertimbangkan jumlah row yang memiliki *missing value* tergolong cukup sedikit. Berikut merupakan *syntax* yang digunakan untuk menangani permasalahan *missing value* terkait:

```
# 1. menangani missing value fitur 'agent'
data['agent'].fillna(0, inplace=True)

# 2. menangani missing value fitur 'company'
data['company'].fillna(0, inplace=True)

# 3. menangani missing value fitur 'country' & 'children'
data.dropna(inplace=True)
```

4.2.2.2. Penanganan *outlier*

Terdapat 2 (dua) *outlier* pada fitur 'ADR' yang masing-masing bernilai -6,38 dan 5400. Secara logis, bisa dianggap ADR tidak mungkin bernilai negatif yang mana dapat dipastikan bahwa data pertama memang benar-benar *outlier*. Sementara untuk data sebesar 5400 memiliki ukuran yang terlampau tinggi dari median yang hanya sebesar 95.Maka dari itu, dapat diasumsikan kedua data tersebut merupakan *outlier data* yang muncul akibat kesalahan imputasi data. Apabila tidak dihilangkan, data *outlier* akan menjadi *noise* yang akan menurunkan performa dari model.

Proses penangan kedua data *outlier* tersebut dilakukan dengan melakukan penghapusan *row data* terkait. Proses penghapusan baris data terkait dapat dilakukan dengan menggunakan *syntax* sebagai berikut:

```
# Penanganan outlier fitur 'ADR'
data.drop(data[data['adr']>1000].index, inplace=True)
data.drop(data[data['adr']<0].index, inplace=True)
```

4.2.2.3. Pengubahan *data type*

Bentuk dari penanganan *data pre-processing* lainnya adalah dengan mengubah *data type*. Pada kasus kali ini, dilakukan pengubahan tipe data dari fitur 'hotel' yang mana merupakan fitur yang bersifat kategorik dengan 2 (dua) jenis kategori yaitu 'resort hotel' dan 'city hotel'. Dengan tujuan untuk memudahkan proses pemodelan, dilakukan pengubahan tipe data menjadi bentuk *boolean*. Selain kasus fitur 'hotel' masih ada beberapa kasus fitur lain yang perlu untuk melalui proses *encoding,* akan tetapi hal tersebut akan dilakukan setelah proses *feature engineering* karena ada beberapa fitur yang dibutuhkan untuk melakukan proses tersebut. Berikut merupakan *syntax* yang digunakan untuk melakukan pengubahan tipe data:

```
# mengubah tipe data fitur 'hotel'
data['hotel'] = data['hotel']=="Resort Hotel"
```

4.2.2.4. Penanganan *imbalance* data

Seperti yang sebelumnya telah dipaparkan pada tahap identifikasi karakteristik data, didapati bahwa terdapat ketidakseimbangan pada *dataset* yang digunakan. Ketidakseimbangan *dataset* pada kasus klasifikasi biner adalah suatu kondisi dimana proporsi jumlah data yang memiliki hasil akhir variabel prediktif bernilai 0 (nol) dengan data yang memiliki hasil akhir bernilai 1 (satu) tidak seimbang (tidak 1 banding 1). Pada kasus terburuk, data yang *imbalance* akan menyebabkan model lebih condong untuk memprediksi ke suatu kelas tertentu dan menyebabkan akurasi dari model menjadi buruk.

Pada kasus kali ini, didapati bahwa proporsi perbandingan antara data yang memiliki hasil akhir prediktif bernilai 0 (nol) dengan data yang memiliki hasil akhir bernilai 1 (satu) senilai 5 (lima) berbanding 3 (tiga). Proporsi tersebut dapat digolongkan sebagai *imbalance* data tingkat rendah. Dikarenakan keadaan data yang masih dalam proporsi *imbalance* tingkat rendah, maka diputuskan untuk tidak melakukan penanganan lebih lanjut pada data, sebagai gantinya akan digunakan *evaluation metrics* yang lebih mengakomodasi ketidak seimbangan data, pada kasus ini akan digunakan *f1 score*.

4.2.2.5. Feature engineering & data transformation

Teknik rekayasa fitur atau *feature engineering*, merupakan salah satu topik informal yang sering kali tidak ke buku *machine learning*, padahal topik ini merupakan salah satu kunci penting kesuksesan model (Brownlee, 2020). Beberapa contoh dari rekayasa fitur dalam analitika data adalah seperti: mendekomposisi fitur kategorik, mendekomposisi fitur tanggal (*datetime*), dan lain-lain.

Pada kasus kali ini, dilakukan sebanyak 12 (dua belas) bentuk *feature engineering*. Berikut merupakan pemaparan untuk masing-masing fitur yang telah berhasil direkayasa:

Tabel 4 pemaparan teknis untuk masing-masing feature engineering

No	Nama fitur	***Keterangan***
1	'stay_days'	Merupakan fitur penjumlahan dari 'stays_in_weekend _nights' dan 'stays_in_week_nights'. Penjumlahan dilakukan dengan mempertimbangkan bentuk persebaran dan korelasi data terhadap variabel target yang sekilas berdistribusi mirip satu sama lain. Tujuan dari penggabungan kedua fitur tersebut adalah untuk menurunkan tingkat kompleksitas model dan menurunkan redundansi data sejenis.
2	'children_babies'	Merupakan fitur penjumlahan dari 'children' dan 'babies'. Penjumlahan dilakukan dengan mempertimbangkan bentuk persebaran dan korelasi data terhadap variabel target yang sekilas berdistribusi mirip satu sama lain. Tujuan dari penggabungan kedua fitur tersebut adalah untuk menurunkan tingkat kompleksitas model dan menurunkan redundansi data sejenis.
3	'country_portugal' & 'benua'	Merupakan fitur yang berasal dari fitur 'country'. Mengelompokkan negara-negara yang berada pada fitur 'country' menjadi per benua. Tujuannya adalah udh menurunkan tingkat kompleksitas model. Mayoritas semua negara dapat dikelompokkan berdasarkan benuanya, akan tetapi terdapat beberapa kasus seperti AQ, UM, dan TF yang tidak terkategorisasi dalam benua manapun. Terdapat pengecualian pengelompokkan data berdasarkan benua untuk negara portugal dikarenakan setelah diamati pada tahapan sebelumnya, didapati bahwa pesanan hotel dalam negeri (Portugal) memiliki polanya tersendiri.
4	'same_room'	Merupakan fitur penanda apakah suatu order memiliki nilai fitur 'reserved_room_type' dengan 'assigned_room_type' memiliki nilai yang sama.
5	'ever_change_book'	Merupakan fitur penanda apakah suatu order pernah disunting atau diganti sebelumnya.
6	'company_book'	Merupakan fitur penanda apakah suatu order dipesan oleh suatu instansi atau perusahaan tertentu.
7	'ever_wait'	Merupakan fitur penanda apakah suatu order pernah masuk ke dalam *waiting list* atau tidak.
8	'parking_space_request'	Merupakan fitur penanda apakah suatu order memiliki nilai fitur 'parking_space' setidaknya sebanyaknya 1 (satu).
9	'use_agent'	Merupakan fitur penanda apakah suatu order melalui suatu agen atau instansi tertentu.
10	'market_segment' masking	Merupakan proses pengelompokkan beberapa jenis *market segment* sebelum melalui proses *encoding*. Pengelompokkan akan dilakukan dari yang semula berjumlah 7 (tujuh) kategori, hingga menjadi 3 (tiga) kelompok saja, yaitu:

No	Nama fitur	*Keterangan*
		• MS_TA/TO → merupakan kategori gabungan dari 'online TA' dan 'offline TA/TO' • MS_Groups → merupakan perubahan nama dari kategori 'Groups' • MS_Lain → merupakan kategori gabungan dari 'Direct', 'Corporate', 'Complementary', dan 'Aviation'. Proses penggabungan mempertimbangkan karakteristik dan korelasi data dengan variabel target. Detail penjelasan mengenai masing-masing karakteristik dari ketiga kelompok bersangkutan, telah terdapat pada *notebook* yang dilampirkan.
11	'assigned_room_type' masking	Merupakan proses pengelompokkan beberapa jenis *room type* sebelum melalui proses *encoding*. Pengelompokkan akan dilakukan dari yang semula berjumlah 12 (dua belas) kategori, hingga menjadi 4 (tiga) kelompok saja yang meliputi 3 kelompok eksisting ('I', 'K', dan 'A') dan kelompok gabungan 'Room_lain'. Proses penggabungan mempertimbangkan karakteristik dan korelasi data dengan variabel target. Detail penjelasan mengenai masing-masing karakteristik dari ketiga kelompok bersangkutan, telah terdapat pada *notebook* yang dilampirkan.
12	one hot encoding untuk fitur-fitur kategorik	Merupakan salah satu bentuk pengaplikasian pengubahan tipe data yang bertujuan supaya data dapat diproses dalam model yang akan digunakan. Beberapa fitur yang mengalami proses encoding diantaranya: • 'meal', • 'continent', • 'distribution_channel', • 'customer_type', • 'deposit_type', • 'market_segment', • 'assigned_room_type' Setelah melakukan *encoding* untuk masing-masing fitur terkait, proses *data pre-processing* ditutup dengan menghapus fitur-fitur yang dirasa tidak diperlukan ataupun tidak dapat digunakan dalam proses pemodelan selanjutnya. Detail mengenai fitur-fitur apa saja yang dihapus beserta *syntax* penghapusan sudah terlampir pada *notebook*.

Proses pelaksanaan *feature engineering* dilakukan secara langsung pada *notebook* menggunakan bahasa pemrograman Python. Berikut merupakan *syntax* yang digunakan pada setiap pelaksanaan *feature engineering* yang telah diapaparkan pada Tabel 4:

Tabel 5 syntax yang digunakan untuk melakukan proses feature engineering

No	Nama fitur	*Syntax*
1	'stay_days'	`data['stay_days'] = data['stays_in_weekend_nights'] + data['stays_in_week_nights']`
2	'children_babies'	`data['children_babies'] = data['children'] + data['babies']`

3	'country_portugal' & 'benua'	`# ubah semua country code menjadi kode benua kecuali untuk portugal`
		`data['continent'] = (data.loc[data['country']!='PRT', ['country']])['country'] .apply(convert_benua)`
		
		`# rename portugal`
		`data['continent'] .fillna("country_portugal", inplace=True)`
		
		`# nanti akan dilakukan proses encoding`

4	'same_room'	`#same_room` `data['same_room'] = data['reserved_room_type'] == data['assigned_room_type']`
5	'ever_change_book'	`data['ever_change_book'] = data['booking_changes'] != 0`
6	'company_book'	`data['company_book'] = data['company']!=0`
7	'ever_wait'	`data['ever_wait'] = data['days_in_waiting_list']!=0`
8	'parking_space_request'	`data['parking_space_request'] = data['required_car_parking_spaces'] != 0`
9	'use_agent'	`data['use_agent'] = data['agent'] != 0`

10	'market_segment' masking	`# MS_TA/TO`
		`data.loc[data['market_segment'] .isin(['Online TA','Offline TA/TO']), 'market_segment'] = "MS_TA/TO"`
		
		`# MS_Lain`
		`data.loc[data['market_segment'] .isin(['Direct','Corporate', 'Complementary','Aviation']), 'market_segment'] = "MS_Lain"`
		
		`# MS_Groups`

No	Nama fitur	*Syntax*
		`data.loc[data['market_segment'] == 'Groups', 'market_segment'] = "MS_Groups"`

11	'assigned_room_type' masking	`# Room_I`
		`data.loc[data['assigned_room_type'] == 'I', 'assigned_room_type'] = "Room_I"`
		
		`# Room_K`
		`data.loc[data['assigned_room_type'] == 'K', 'assigned_room_type'] = "Room_K"`
		
		`# Room_A`
		`data.loc[data['assigned_room_type'] == 'A', 'assigned_room_type'] = "Room_A"`
		
		`# Room_Lain`
		`data.loc[~data['assigned_room_type'] .isin(['I','K','A']), 'assigned_room_type'] = "Room_Lain"`

12	one hot encoding untuk fitur-fitur kategorik	`encode_list = ['meal', 'continent', 'distribution_channel', 'customer_type', 'deposit_type','market_segment', 'assigned_room_type']` `data = pd.get_dummies(data=data, columns = encode_list)`

4.2.3. Pemodelan data

Proses rekayasa fitur yang telah dipaparkan sebelumnya akan menjadi akhir penutup dari tahapan *data pre-processing*. Proses pemodelan berlanjut ke teknis langsung dari *pemodelan data*. Proses pemodelan data akan dibagi menjadi 5 (lima) tahapan besar, yang terdiri dari: *splitting dataset, testing model, feature selection, hyperparameter tuning,* dan *model stacking*.

4.2.3.1. Splitting dataset

Tahapan pemodelan data dimulai dari proses *splitting dataset*. Proses *splitting dataset* adalah proses pemisahan *dataset* menjadi beberapa bagian yang kemudian nantinya akan digunakan sebagai salah satu sumber proses pemodelan. Pada penelitian kali ini, **tidak dilakukan pemisahan antara data *train* dengan data *test*** dikarenakan **proporsi antara fitur dengan ukuran dataset** dimana jumlah fitur relatif tergolong cukup banyak untuk ukuran dataset yang dimiliki. Jika dipaksakan untuk melakukan pemisahan, maka dikhawatirkan akan terjadinya kasus *underfit*. Sebagai gantinya, proses validasi akan dilakukan dengan teknik *cross validation* seperti yang pernah dijelaskan pada landasan teori.

Meskipun tidak terjadi pemisahan antara data *train* dengan data *test*, proses pemisahan tetap dilaksanakan khususnya untuk memisahkan antara fitur-fitur prediktif (X) dengan fitur target yang akan diprediksi (y). Proses pemisahan dilakukan menggunakan *syntax* sebagai berikut:

```
x = data_no_agent.drop(['is_canceled'], axis=1)
y = data_no_agent['is_canceled']
```

4.2.3.2. Testing model

Proses selanjutnya adalah mencoba untuk menyocokan data ke dalam model-model yang dirasa cukup potensial dalam mengatasi permasalahan klasifikasi pembatalan pesanan hotel. Pada kasus kali ini, dipilih 4 (empat) calon model yang dirasa cukup potensial, diantaranya: *logistic regression, random forest, XGB Classifier,* dan *CatBoost Classifier*. Model *logistic regression* dan *random forest* dipilih karena terkenal dengan kemudahan pemakaiannya untuk kasus klasifikasi, yang mana tidak terlalu membutuhkan *hyperparameter tuning* untuk mendapatkan hasil yang optimal. Sementara *XGB Classifier* dan *CatBoost Classifier* dipilih dengan mempertimbangkan kemudahan pemakaian dan hasil yang rata-rata tergolong cukup efektif untuk menghadapi berbagai jenis kasus klasifikasi.

Proses pemodelan masih dilakukan dengan media yang sama yaitu *notebook* berbahasa Python dengan bantuan beberapa *package* yang dibutuhkan untuk proses pemodelan pada umumnya. Berikut merupakan *syntax* dari proses pemanggilan *package-package* yang digunakan pada kasus pemodelan kali ini:

```
#Define cross validation indices generator
from sklearn.model_selection import KFold
kf = KFold(n_splits=5, random_state=1, shuffle=True)

#define package keperluan lainnya
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import RFECV
from sklearn.feature_selection import SelectKBest, chi2,
f_classif

#Define evaluation metrics yang digunakan, yaitu f1_score
from sklearn.metrics import f1_score

#define model yang digunakan
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier

#define keperluan hyperparameter tuning
from sklearn.model_selection import GridSearchCV
```

Model pertama yang dicoba adalah model *logistic regression*. Model *logistic regression* seperti yang telah disampaikan pada landasan teori merupakan salah satu jenis *linear model* yang merupakan pengembangan dari *linear regression*. Ketika dicoba, model *logistic regression* menghasilkan *f1 score* sebesar 0,708. Berikut merupakan *syntax* yang digunakan untuk melakukan proses pemodelan tersebut:

```
x_lr = x.copy()
y_lr = y.copy()
```

```
lr = LogisticRegression(verbose=0)
score_lr = cross_val_score(lr, x_lr, y_lr, cv=kf,
scoring='f1')
print(score_lr)
print(f"Akurasi Model: {np.mean(score_lr)}")
```

Model kedua yang dicoba adalah model *random forest*. Model *random forest* seperti yang telah disampaikan pada landasan teori merupakan gabungan dari berbagai *decision tree models*. Ketika dicoba, model *random forest* menghasilkan *f1 score* sebesar 0,818. Berikut merupakan *syntax* yang digunakan untuk melakukan proses pemodelan tersebut:

```
x_rfc = x.copy()
y_rfc = y.copy()

rlc = RandomForestClassifier(random_state=1, verbose=0)
score_rlc = cross_val_score(rlc, x_rfc, y_rfc, cv=kf,
scoring='f1')
print(score_rlc)
print(f"Akurasi Model: {np.mean(score_rlc)}")
```

Model ketiga yang dicoba adalah model *XGBoost Classifier*. Model *XGBoost Classifier* seperti yang telah disampaikan pada landasan teori merupakan salah satu jenis model yang berbasis pada teknik *boosting*. Ketika dicoba, model *XGBoost Classifier* menghasilkan *f1 score* sebesar 0,810. Berikut merupakan *syntax* yang digunakan untuk melakukan proses pemodelan tersebut:

```
x_xgb = x.copy()
y_xgb = y.copy()

xgb = XGBClassifier(random_state=1, verbose=0)
score_xgb = cross_val_score(xgb, x_xgb, y_xgb, cv=kf,
scoring='f1')
print(score_xgb)
print(f"Akurasi Model: {np.mean(score_xgb)}")
```

Model ketiga yang dicoba adalah model *CatBoost Classifier*. Model *CatBoost Classifier* seperti yang telah disampaikan pada landasan teori merupakan salah satu jenis model yang berbasis pada teknik *boosting*. Ketika dicoba, model *CatBoost Classifier* menghasilkan *f1 score* sebesar 0,814. Berikut merupakan *syntax* yang digunakan untuk melakukan proses pemodelan tersebut:

```
x_cbc = x.copy()
y_cbc = y.copy()

cbc = CatBoostClassifier(random_state=1, verbose=0)
score_cbc = cross_val_score(cbc, x_cbc, y_cbc, cv=kf,
scoring='f1')
print(score_cbc)
print(f"Akurasi Model: {np.mean(score_cbc)}")
```

Berdasarkan hasil yang sudah diperoleh dari setiap percobaan model yang potensial, dapat ditarik 2 (dua) poin kesimpulan. Poin kesimpulan pertama adalah ditemukan bahwa *logistic regression* ternyata kurang cocok untuk digunakan untuk melakukan klasifikasi permasalahan terkait dimana tampak bahwa hasil akurasi yang dihasilkan dari model *logistic regression* cenderung jauh lebih rendah daripada model-model lainnya. Kesimpulan kedua adalah penerimaan 3 (tiga) model potensial yang akan lanjut ke tahap berikutnya, yaitu: *random forest, XGBoost Classifier,* dan *CatBoost Classifier*.

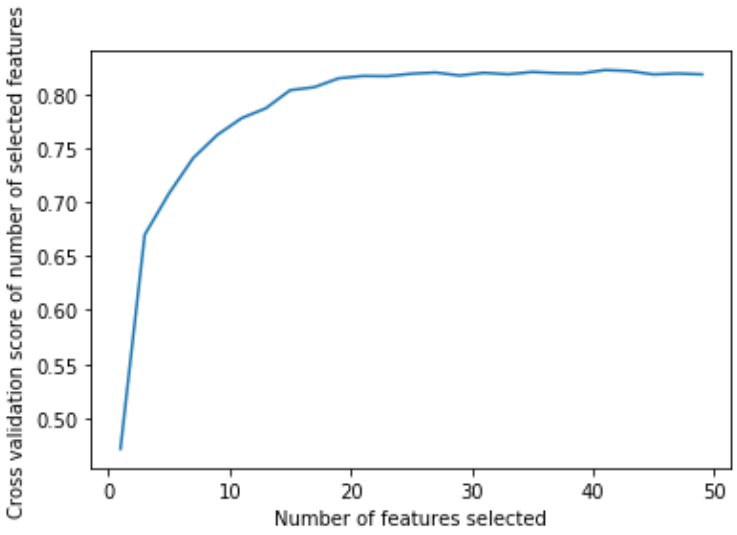
4.2.3.3. Feature selection

Setelah berhasil memperkirakan hasil kasar pemodelan dari setiap model potensial yang dipilih sebelumnya, proses selanjutnya adalah tahapan *feature selection*. Seperti yang sudah dijelaskan pada landasan teori sebelumnya, pemilihan fitur diperlukan untuk mengeliminasi fitur-fitur yang bersifat *noise* yang mampu berdampak secara negatif terhadap hasil akhir prediksi.

Pada kasus penelitian tentang pembangunan model prediksi klasifikasi pembatalan pesanan hotel, terdapat total 48 (empat puluh delapan) fitur yang akan digunakan sebagai basis data pemodelan. Proses pengeliminasian fitur akan diprioritaskan untuk dilakukan dengan menggunakan metode RFECV (*recursive feature elimination cross validation*). Pada beberapa kasus seperti model XGBoost, proses RFECV tidak *feasible* untuk dilakukan karena beberapa alasan yang akan dijelaskan berikutnya secara lebih detail.

Pada kasus seleksi fitur pada model *random forest*, digunakan teknik RFECV. Proses iterasi menghasilkan bahwa fitur optimal yang sebaiknya digunakan pada model *random forest* adalah sebanyak 40 (empat puluh) fitur yang detailnya dapat dilihat secara langsung pada *notebook* yang dilampirkan. Dengan hanya menggunakan 40 (empat puluh) fitur terpilih saja, didapatkan akurasi sebesar 0,821 atau terdapat peningkatan akurasi sebesar 0,35%. Berikut merupakan *syntax* yang digunakan untuk melakukan proses seleksi fitur menggunakan RFECV:

```
X_rf = X.copy()
y_rf = y.copy()
rfecv_rf = RFECV(RandomForestClassifier(random_state=1,
n_estimators = 10), scoring = 'f1', step=2, cv=kf)
rfecv_rf.fit(X_rf, y_rf)

print('Optimal number of features :', rfecv_rf.n_features_)
print('Best features :', X_rf.columns[rfecv_rf.support_]) #
rfecv.support --> ambil indices


plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score of number of selected
features")
plt.plot(range(1, 2*len(rfecv_rf.grid_scores_)+ 1, 2),
rfecv_rf.grid_scores_)
plt.show()
```



Figur 9 grafik perbandingan akurasi model random forest terhadap jumlah fitur yang dipilih

Pada kasus seleksi fitur pada model *XGBoost Classifier*, mula-mula digunakan teknik RFECV. Akan tetapi dikarenakan modul *xgboost* tidak mengakomodasi sistem *select from model* yang diusung oleh fungsi RFECV sklearn, maka proses RFECV tidak dapat diberlakukan. Sebenarnya terdapat solusi untuk dapat menggunakan RFECV pada pemodelan *XGBoost*, yaitu dengan menurunkan versi XGBoost ke versi di bawah 1, akan tetapi hal tersebut ditakutkan akan mempengaruhi performansi model, sehingga diputuskan untuk tidak melakukan penanganan tersebut.

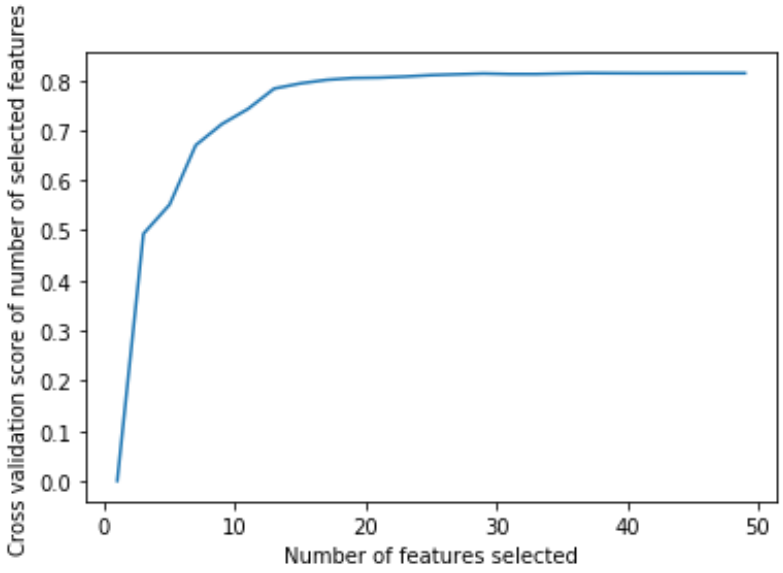
Sebagai ganti teknik RFECV yang gagal diaplikasikan pada model *XGBoost Classifier*, dilakukan proses seleksi fitur dengan menggunakan teknik lainnya yang disebut SelectKBest. Penjelasan detail mengenai SelectKBest sudah dijelaskan pada landasan teori. Pada proses seleksi fitur, digunakan 2 (dua) jenis *score function*, yaitu: chi2 atau *chi-squared* & *f_classif*. Akan tetapi setelah dicoba melakukan reduksi fitur, ternyata didapati bahwa skor akurasi cenderung turun bersamaan dengan penurunan volume fitur. Maka dari itu, diputuskan untuk tidak melakukan *feature selection* pada model *XGBoost Classifier*. Detail mengenai hasil *feature selection* terdapat pada *notebook* yang dilampirkan. Berikut merupakan *syntax* yang digunakan untuk melakukan seleksi fitur menggunakan teknik SelectKBest:

```
x_xgb_fs = x.copy().iloc[:,SelectKBest(chi2,
k=35).fit(x.copy(), y.copy()).get_support()].to_numpy()
y_xgb_fs = y.copy()

xgb = XGBClassifier(random_state=1, verbose=0)

score_xgb_fs = cross_val_score(xgb, x_xgb_fs, y_xgb_fs, cv=kf,
scoring='f1')
print(score_xgb_fs)
print(f"Akurasi Model: {np.mean(score_xgb_fs)}")

x_xgb_fs = x.copy().iloc[:,SelectKBest(f_classif,
k=40).fit(x.copy(), y.copy()).get_support()].to_numpy()
y_xgb_fs = y.copy()
```

```
xgb = XGBClassifier(random_state=1, verbose=0)

score_xgb_fs = cross_val_score(xgb, x_xgb_fs, y_xgb_fs, cv=kf,
scoring='f1')
print(score_xgb_fs)
print(f"Akurasi Model: {np.mean(score_xgb_fs)}")
```

Pada kasus seleksi fitur pada model *CatBoost Classifier*, digunakan teknik RFECV. Proses iterasi menghasilkan bahwa fitur optimal yang sebaiknya digunakan pada model *CatBoost Classifier* adalah sebanyak 36 (tiga puluh enam) fitur yang detailnya dapat dilihat secara langsung pada *notebook* yang dilampirkan. Dengan hanya menggunakan 36 (tiga puluh enam) fitur terpilih saja, didapatkan akurasi sebesar 0,814, tidak terdapat pertambahan akurasi secara signifikan, akan tetapi proses seleksi fitur ini tetap dilanjutkan dalam rangka menurunkan kompleksitas dari model. Berikut merupakan *syntax* yang digunakan untuk melakukan proses seleksi fitur menggunakan RFECV:

```
x_cbc_fs = x.copy()
y_cbc_fs = y.copy()

cbc = CatBoostClassifier(random_state=1, verbose = 0)
rfecv_cbc = RFECV(cbc, scoring = 'f1', step = 2, cv=kf)
rfecv_cbc.fit(x_cbc_fs, y_cbc_fs)

print('Optimal number of features :', rfecv_cbc.n_features_)
print('Best features :', x_cbc_fs.columns[rfecv_cbc.support_])
# rfecv.support --> ambil indices

plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score of number of selected
features")
plt.plot(range(1, 2*len(rfecv_cbc.grid_scores_) + 1, 2),
rfecv_cbc.grid_scores_)
plt.show()
```



Figur 10 grafik perbandingan akurasi model CatBoost Classifier terhadap jumlah fitur yang dipakai

4.2.3.4. Hyperparameter Tuning

Proses selanjutnya setelah usai memilih fitur-fitur pilihan untuk masing-masing model adalah proses *hyperparameter tuning*. Definisi *hyperparameter tuning* sudah dipaparkan sebelumnya pada landasan teori. Proses *tuning* hanya ditujukan khusus untuk model *random forest* yang sementara memiliki hasil akurasi terbaik. Pada proses *tuning* model *random forest*, dilakukan *tuning* secara otomatis dengan teknik Grid Search. *Hyperparameter* yang akan menjalani proses *tuning* adalah *hyperparameter* 'max_features' dan 'n_estimators'.

'max_features' merupakan *hyperparameter* yang akan mempengaruhi limit pengambilan fitur yang akan dipertimbangkan pada setiap splitting, khususnya dalam konteks melakukan dekorelasi antar *decision tree* yang akan dibuat. Nilai *default* dari 'max_features' adalah senilai akar kuadrat dari jumlah fitur. Semakin besar nilai 'max_features' maka semakin kecil variansi yang akan terjadi antar setiap *decision tree* yang akan dibuat. Sebaliknya, apabila 'max_features' semakin kecil, maka variansi akan menjadi semakin tinggi, hingga pada beberapa kasus terjadi *underfit*.

'n_estimators' merupakan *hyperparameter* yang menentukan banyaknya *decision tree* yang akan dibuat dan nantinya digabungkan menjadi model *random forest*. Secara umum, jika 'n_estimators' bernilai terlalu kecil, maka akan berpeluang terjadinya underfit, sebaliknya jika terlalu besar, maka waktu proses dan *resource* yang diperlukan untuk memodelkan data menjadi meningkat. Berikut merupakan *syntax* yang digunakan untuk melakukan *hyperparameter tuning* menggunakan teknik Grid Search:

```
x_rf_ht = rfecv_rf.transform(x.copy())
y_rf_ht = y.copy()

rf = RandomForestClassifier(random_state=1) #bootstrap = True,
criterion = 'gini'
```

```
param_grid_rf = {
    'max_features':[8,10,12],
    'n_estimators':[50,200,500]
}

gs_rf = GridSearchCV(rf, param_grid_rf, scoring='f1', cv=kf,
verbose=0)
gs_rf.fit(x_rf_ht,y_rf_ht)

print(f"best parameters: {gs_rf.best_params_}")
print(f"best score: {gs_rf.best_score_}")
```

Setelah melakukan iterasi untuk masing-masing kombinasi *hyperparameter* pada *Grid Search*, didapatkan bahwa *hyperparameter* optimal untuk model *random forest* pada kasus ini adalah 'max_features' senilai 10 dan 'n_estimators' senilai 200. Kombinasi dari kedua *hyperparameter* tersebut beserta tahapan sebelumnya (*feature selection*) menghasilkan akurasi model *random forest* sebesar 0,8356.

4.2.3.5. Model Stacking

Proses terakhir dari tahapan pemodelan data adalah *model stacking*. Seperti yang telah dijelaskan pada landasan teori, pada *model stacking* menggunakan beberapa model yang berbeda. Pada kasus ini diputuskan untuk menggunakan 3 (tiga) model optimal yang sebelumnya sudah melewati proses *feature selection* dan *hyperparameter tuning*, yaitu: *random forest*, *XGBoost*, dan *CatBoost*. Sementara untuk meta model yang digunakan adalah *logistic regression* yang tidak memerlukan *hyperparameter tuning* lebih lanjut. Berdasarkan kombinasi dari berbagai model tersebut, didapatkan hasil akhir skor akurasi 0,8354. Berikut merupakan *syntax* yang digunakan untuk melaksanakan *model stacking*:

```
x_stack = x.copy()
y_stack = y.copy()

# dataset splitting
xtrain, xval, ytrain, yval = train_test_split(x_stack,
y_stack, random_state=1, test_size = 0.25, shuffle=True)

rfc = RandomForestClassifier(n_estimators=200,
max_features=10, random_state=1, verbose=0)
xgb = XGBClassifier(random_state=1, verbose = 0)
cbc = CatBoostClassifier(random_state=1, verbose = 0)
meta_lr = LogisticRegression()

# Model-1: random forest
x_stack_rfc = rfecv_rf.transform(xtrain)
y_stack_rfc = ytrain
rfc.fit(x_stack_rfc, y_stack_rfc)

# Model-2: xgboost
x_stack_xgb = xtrain
y_stack_xgb = ytrain
xgb.fit(x_stack_xgb, y_stack_xgb)
```

```
# Model-3: catboost
x_stack_cbc = rfecv_cbc.transform(xtrain)
y_stack_cbc = ytrain
cbc.fit(x_stack_cbc, y_stack_cbc)

# Meta Model: Logistic Regression
xval_meta = pd.DataFrame({
    'rfc_score':rfc.predict(rfecv_rf.transform(xval)),
    'xgb_score':xgb.predict(xval),
    'cbc_score':cbc.predict(rfecv_cbc.transform(xval))
})
yval_meta = yval
score_stacked = cross_val_score(meta_lr, xval_meta, yval_meta,
cv=kf, scoring='f1')
print(score_stacked)
print(f"Akurasi Model: {np.mean(score_stacked)}")
```

4.3. Perumusan Rekomendasi Tindak Lanjut

Setelah berhasil melakukan pemodelan data, tahapan selanjutnya adalah menggali informasi-informasi yang dapat diperoleh melalui percobaan yang sudah berhasil dilakukan. Penggalian informasi secara umum akan dibagi menjadi 2 (dua) jenis, yaitu penggalian informasi mengenai model yang terbaik dan perihal fitur-fitur relevan yang berperan aktif dalam proses pemodelan.

4.3.1. Penentuan model terbaik

Proses penentuan model terbaik mempertimbangkan berbagai aspek-aspek yang terdapat pada setiap model. Hal-hal yang perlu dipertimbangkan dalam penentuan model terbaik diantara lain: akurasi dari model itu sendiri, lama waktu *fitting*, lama waktu perhitungan prediksi, *computing resource* yang diperlukan, dan aspek lainnya.

Apabila ditinjau dari aspek akurasi tiap model, terdapat 2 (dua) calon potensial yang dapat dijadikan sebagai model terbaik, yaitu: model *random forest* paska *hyperparameter tuning* dan *stacked model random forest, xgboost,* dan *CatBoost* dengan meta model *logistic regression*. Keduanya memiliki akurasi yang hampir mirip yaitu berada pada kisaran angka 0,83. Akan tetapi, model *random forest* paska *hyperparameter tuning* memiliki akurasi yang sedikit lebih baik jika dibandingkan dengan model gabungan.

Apabila ditinjau dari lama waktu *fitting*, lama waktu perhitungan prediksi, dan *computer resource*, tentunya bisa dipastikan bahwa model *random forest* paska *hyperparameter tuning* akan menggunakan *resource* yang lebih sedikit jika dibandingkan dengan *tacked model random forest, xgboost,* dan *CatBoost* dengan meta model *logistic regression.* Berdasarkan fakta tersebut, maka dapat ditarik kesimpulan bahwa **model terbaik** yang sebaiknya digunakan dalam proses klasifikasi pembatalan pesanan adalah model ***random forest*** dengan *hyperparameter* 'max_features' sebesar 10, dan 'n_estimators' sebesar 200.

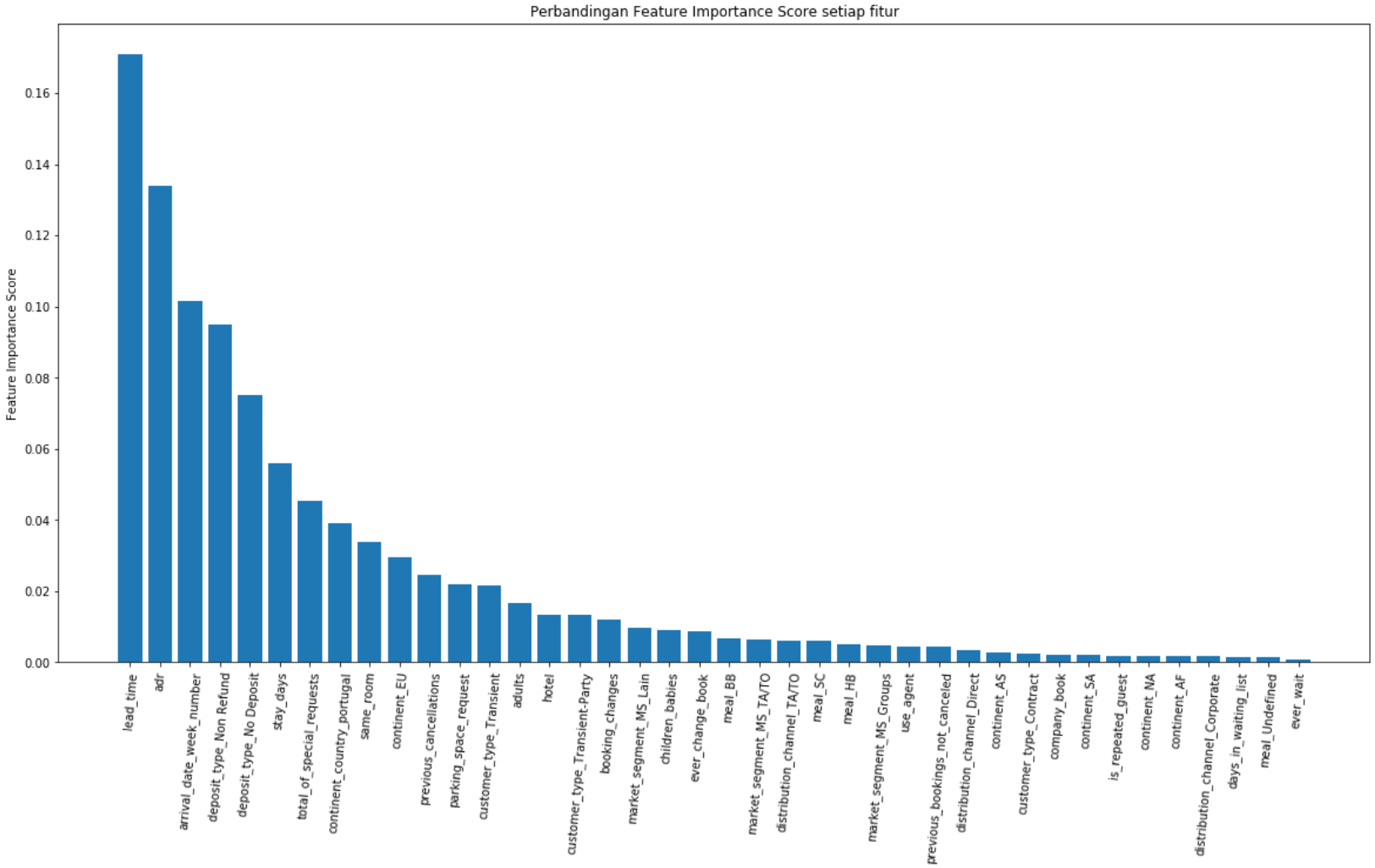
4.3.2. Penentuan fitur-fitur yang relevan

Setelah berhasil menentukan model terbaik yang akan diusulkan untuk implementasi selanjutnya kepada stakeholder, tahapan selanjutnya adalah penentuan fitur-fitur relevan. Proses penentuan fitur-fitur relevan menggunakan bantuan *feature importance score* yang diperoleh berdasarkan rata-rata *gini score* yang dihasilkan dalam setiap *decision tree estimator* yang terdapat pada model *random forest*. Berikut merupakan *syntax* untuk mengetahui besar *feature importance* dari masing-masing fitur:

```
rlc.fit(x_rfc_fs, y_rfc_fs)
fi = {
    "feature": x.iloc[:,rfecv_rf.support_].columns,
    "feature importance": rlc.feature_importances_
}
fi_df = pd.DataFrame(fi)
fi_df.sort_values('feature importance', ascending=False,
inplace=True)
fi_df
```

Dalam rangka mempermudah proses pengamatan dan analisis, data *feature importance* yang telah diperoleh sebelumnya akan divisualisasikan ke dalam bentuk grafik. Figur 11 merupakan grafik perbandingan antara *feature importance score* untuk masing-masing fitur yang terdapat pada proses pemodelan prediksi pembatalan kamar hotel. Berikut merupakan *syntax* yang digunakan untuk melakukan visualisasi terkait:

```
plt.figure(figsize = (20,10))
plt.xticks(rotation = 85)
plt.bar(fi_df['feature'], fi_df['feature importance'])
plt.title("Perbandingan Feature Importance Score setiap
fitur")
plt.ylabel("Feature Importance Score")
```



Figur 11 perbandingan feature importance score untuk masing-masing fitur

Berdasarkan grafik yang telah dipetakan di atas, dapat diperoleh informasi **5 (lima) fitur yang paling berpengaruh yaitu: 'lead_time', 'adr', 'arrival_date_week_number', 'deposit_type_non_refund', dan 'deposit_type_no_deposit'**. Berdasarkan temuan tersebut, dapat dilakukan analisis lebih lanjut yang nantinya akan dipaparkan pada bab selanjutnya.

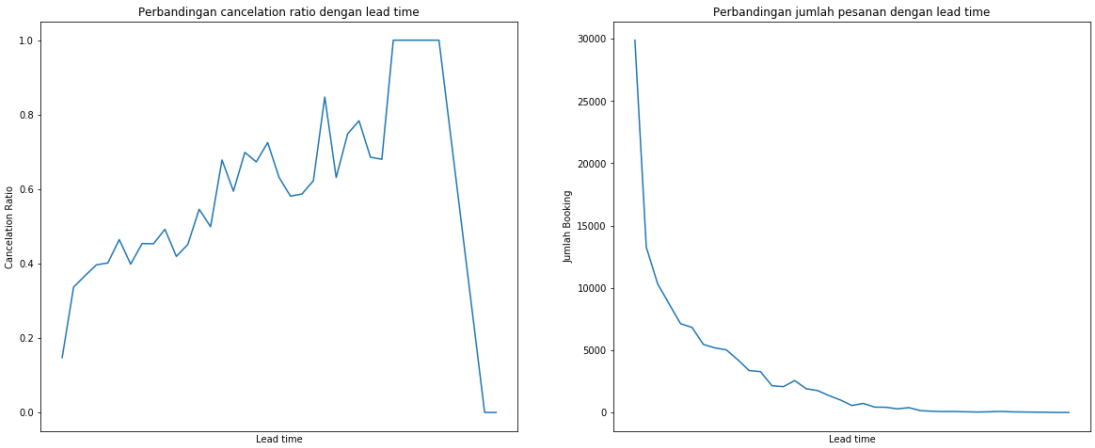
BAB V
ANALISIS MANAJERIAL

5.1. Analisis Fitur 'Lead Time' terhadap Keberhasilan Pemetaan Pembatalan Pesanan Kamar Hotel

Berdasarkan pengelompokkan fitur-fitur paling relevan yang ditinjau dari nilai *feature importance*, didapati bahwa fitur 'Lead Time' merupakan salah satu fitur yang berperan dalam proses klasifikasi apakah suatu pesanan akan dibatalkan atau tidak. Seperti yang telah diinformasikan pada Tabel 3 feature dictionary, didapati bahwa *lead time* merupakan selisih jumlah hari antara tanggal pencatatan data pesanan ke dalam sistem dengan tanggal kedatangan.

Dalam rangka memahami karakteristik fitur 'Lead Time' dengan lebih baik, dilakukan visualisasi data khususnya yang dapat menampilkan perbandingan antara rasio pembatalan dengan lama *lead time*. Selain itu, sebagai informasi tambahan, perlu juga untuk memvisualisasikan perbandingan antara jumlah pesanan dengan lama *lead time*. Berikut merupakan *syntax* yang digunakan untuk melakukan visualisasi terkait:

```
# Perhitungan cancelation ratio per kelas lama lead time
data_lead_time = data[['is_canceled', 'lead_time']].copy()
data_lead_time['lead_time_class'] =
pd.cut(data_lead_time['lead_time'], bins=(1 +
3.3*math.log(len(data_lead_time)))).cat.codes
data_lead_cancel_ratio =
data_lead_time.groupby('lead_time_class')['is_canceled'].mean(
).dropna()

# perhitungan jumlah booking per kelas lama lead time
data_lead_number_book =
data_lead_time.groupby('lead_time_class')['is_canceled'].count
().dropna()

# plotting
fig = plt.figure(figsize = (20,8))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

ax1.plot(data_lead_cancel_ratio.index, data_lead_cancel_ratio)
ax1.set_title("Perbandingan cancelation ratio dengan lead
time")
ax1.set_xticks([])
ax1.set_xlabel("Lead time")
ax1.set_ylabel("Cancelation Ratio")

ax2.plot(data_lead_number_book.index, data_lead_number_book)
ax2.set_title("Perbandingan jumlah pesanan dengan lead time")
ax2.set_xticks([])
ax2.set_xlabel("Lead time")
ax2.set_ylabel("Jumlah Booking")
```



Figur 12 hasil visualisasi fitur lead time

Berdasarkan hasil visualisasi yang terlihat pada Figur 12, didapati bahwa terdapat tren kenaikan rasio pembatalan seiring dengan besarnya *lead time*. Sebaliknya, jumlah pesanan memiliki tren menurun untuk *lead time* yang semakin tinggi. Berdasarkan informasi tersebut, pihak manajemen memiliki beberapa opsi kebijakan yang dapat dipertimbangkan untuk diimplementasikan.

Opsi pertama adalah dengan memberlakukan **sistem maksimal *lead time* pemesanan**. Dampak positif dari adanya mekanisme ini adalah menurunnya rasio pembatalan. Akan tetapi, dampak negatifnya adalah kemungkinan penurunan jumlah pesanan. **Opsi kedua** adalah dengan **memberikan insentif langsung untuk pesanan dengan *lead time* yang rendah**. Pemberian insentif langsung dapat berupa pemberian diskon ataupun promo yang akan mengakibatkan pesanan bergeser ke arah *lead time* yang lebih rendah. **Opsi ketiga** adalah memberikan **insentif tidak langsung untuk pesanan dengan *lead time* yang tinggi**. Bentuk insentif tidak langsung dapat berupa pemberian *cashback* apabila konsumen benar-benar memenuhi dan tidak membatalkan pesanan pada hari kedatangan.

5.2. Analisis Fitur 'ADR' atau Average Daily Rate terhadap Keberhasilan Pemetaan Pembatalan Pesanan Kamar Hotel

Average Daily Rate atau ADR, merupakan salah satu KPI (*key performance indicator*) yang umum digunakan dalam industri perhotelan (Hargrave, 2020). ADR memberikan gambaran dasar mengenai rata-rata seberapa besar *revenue* yang dihasilkan per ruangan atau kamar hotel. Persamaan 3 merupakan persamaan perhitungan untuk ADR.

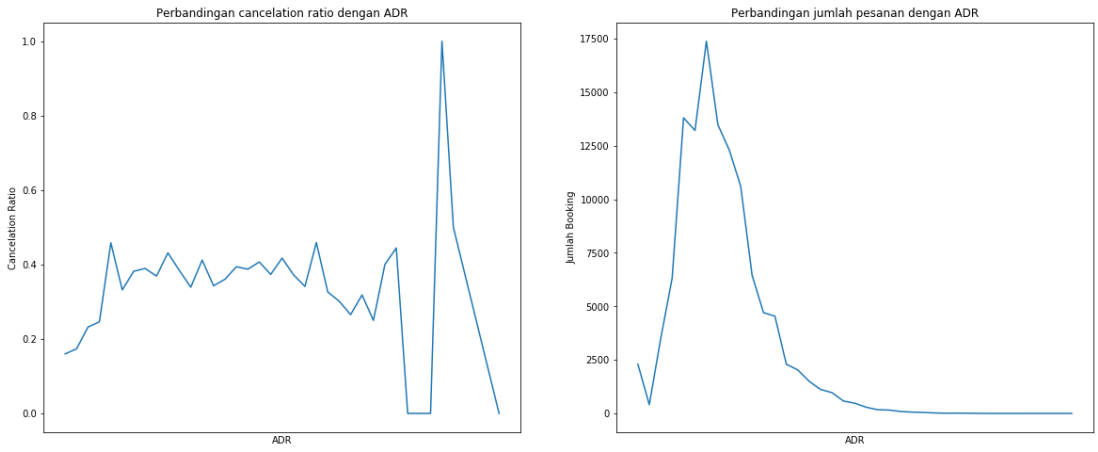
$$Average\ Daily\ Rate = \frac{Rooms\ Revenue\ Earned}{Number\ of\ Rooms\ Sold}$$

Persamaan 3 average daily rate (ADR)

Berdasarkan definisinya, ADR yang tinggi menandakan bahwa pada hari tersebut banyak kamar hotel yang menghasilkan *revenue* besar yang dipesan. Sebaliknya, apabila ADR rendah, hal tersebut menandakan bahwa lebih banyak kamar hotel bertipe *revenue* kecil yang dipesan. Berdasarkan temuan tersebut, akan dilakukan visualisasi data khususnya yang dapat menampilkan perbandingan antara rasio pembatalan dengan besar ADR. Selain itu,

sebagai informasi tambahan, perlu juga untuk memvisualisasikan perbandingan antara jumlah pesanan dengan ADR. Berikut merupakan *syntax* yang digunakan untuk melakukan visualisasi terkait:

```
# Perhitungan cancelation ratio per kelas ADR
data_adr = data[['is_canceled', 'adr']].copy()
data_adr['adr_class'] = pd.cut(data_adr['adr'], bins=(1 +
3.3*math.log(len(data_adr)))).cat.codes
data_adr_cancel_ratio =
data_adr.groupby('adr_class')['is_canceled'].mean().dropna()

# perhitungan jumlah booking per kelas ADR
data_adr_number_book =
data_adr.groupby('adr_class')['is_canceled'].count().dropna()

# plotting
fig = plt.figure(figsize = (20,8))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

ax1.plot(data_adr_cancel_ratio.index, data_adr_cancel_ratio)
ax1.set_title("Perbandingan cancelation ratio dengan ADR")
ax1.set_xticks([])
ax1.set_xlabel("ADR")
ax1.set_ylabel("Cancelation Ratio")

ax2.plot(data_adr_number_book.index, data_adr_number_book)
ax2.set_title("Perbandingan jumlah pesanan dengan ADR")
ax2.set_xticks([])
ax2.set_xlabel("ADR")
ax2.set_ylabel("Jumlah Booking")
```



Figur 13 hasil visualisasi ADR

Berdasarkan hasil visualisasi yang terlihat pada Figur 13, didapati bahwa di awal terdapat tren kenaikan rasio pembatalan seiring dengan besarnya ADR, kemudian tren tersebut hilang pada titik tertentu dan berubah menjadi konstan. Berdasarkan informasi tersebut, pihak manajemen perhotelan tidak perlu terlalu mengkhawatirkan pembatalan

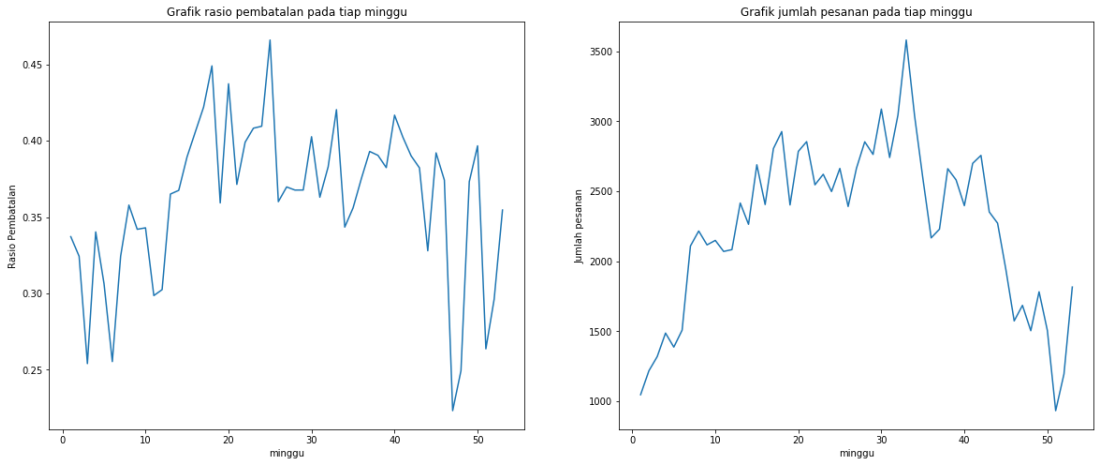
pesanan pada saat ADR rendah dan bisa lebih berfokus untuk meningkatkan ADR. Sebaliknya, ketika ADR semakin tinggi, pihak manajemen perlu semakin berhati-hati untuk menjaga rasio pembatalan supaya tidak terjadi *loss revenue* yang berlebihan.

Bentuk program yang dapat dilakukan oleh pihak manajemen pada **kondisi ADR rendah** misalnya saja dengan **memberikan insentif lebih untuk pemesanan kamar hotel yang memiliki *revenue* tinggi**. Sementara pada kondisi ADR tinggi, pihak manajemen dapat memperhatikan program-program lain yang memerlukan pengembangan.

5.3. Analisis Fitur 'Arrival_date_week_number' terhadap Keberhasilan Pemetaan Pembatalan Pesanan Kamar Hotel

Seperti yang sudah dijelaskan pada Tabel 3 feature dictionary, *arrival date week number* merupakan urutan minggu dari tanggal kedatangan suatu pesanan. Dalam rangka memahami karakteristik fitur 'arrival_date_week_number' dengan lebih baik, dilakukan visualisasi data khususnya perihal perbandingan rasio pembatalan tiap minggu . Berikut merupakan *syntax* yang digunakan untuk melakukan visualisasi terkait:

```
fig = plt.figure(figsize=(20,8))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

ax1.plot(data.groupby('arrival_date_week_number')['is_canceled
'].mean())
ax1.set_title('Grafik rasio pembatalan pada tiap minggu')
ax1.set_xlabel("minggu")
ax1.set_ylabel("Rasio Pembatalan")

ax2.plot(data['arrival_date_week_number'].value_counts().sort_
index())
ax2.set_title('Grafik jumlah pesanan pada tiap minggu')
ax2.set_xlabel("minggu")
ax2.set_ylabel("Jumlah pesanan")
```



Figur 14 hasil visualisasi arrival date week number

Berdasarkan hasil visualisasi yang dapat dilihat pada Figur 14, didapati bahwa rasio pembatalan cenderung tinggi pada pertengahan tahun dan rendah pada awal dan akhir tahun. Hal serupa juga ditemui pada jumlah pesanan yang tinggi pada pertengahan tahun dan rendah pada awal dan akhir tahun, terkecuali pada minggu akhir (minggu ke-54) yang mengalami sedikit lonjakan yang diduga akibat dari liburan akhir tahun dan natal.

Apabila ditinjau dari budaya bepergian yang ada di eropa, menurut data yang diterbitkan oleh **Eurostat**, didapati bahwa dari **tahun 2008 hingga tahun 2018**, terdapat **tren penurunan alasan bepergian akibat bisnis dan terdapat tren kenaikan alasan bepergian akibat keperluan ataupun keinginan pribadi** (Eurostat, 2021). Apabila dikaitkan dengan *dataset* yang digunakan dalam penelitian ini, seperti yang sudah dipaparkan pada subbab sebelumnya, *hotel booking demand datasets* merupakan kompilasi data pesanan hotel di Portugal yang diambil pada rentang waktu Juli 2015 sampai Agustus 2017. Maka bisa disimpulkan bahwa kemungkinan besar **pesanan kamar akan didominasi oleh konsumen yang memiliki alasan untuk bepergian atas kebutuhan atau keinginan pribadi**.

Informasi di atas dapat dikaitkan dengan iklim dan kebiasaan *traveling* yang ada di Portugal. Diketahui **iklim negara Portugal** tergolong mediteranian dengan **4 (empat) jenis musim**, dengan **proporsi musim panas yang lebih banyak di pertengahan tahun**. Sementara jika ditinjau dari kebiasaan ***traveling***, maka kebutuhan rekreasi akibat keinginan pribadi secara logis akan **lebih mungkin untuk terjadi pada musim panas** atau pertengahan tahun. Hal tersebut didukung dengan kondisi geografis Negara Portugal yang **berbatasan langsung dengan Samudra Atlantik**.

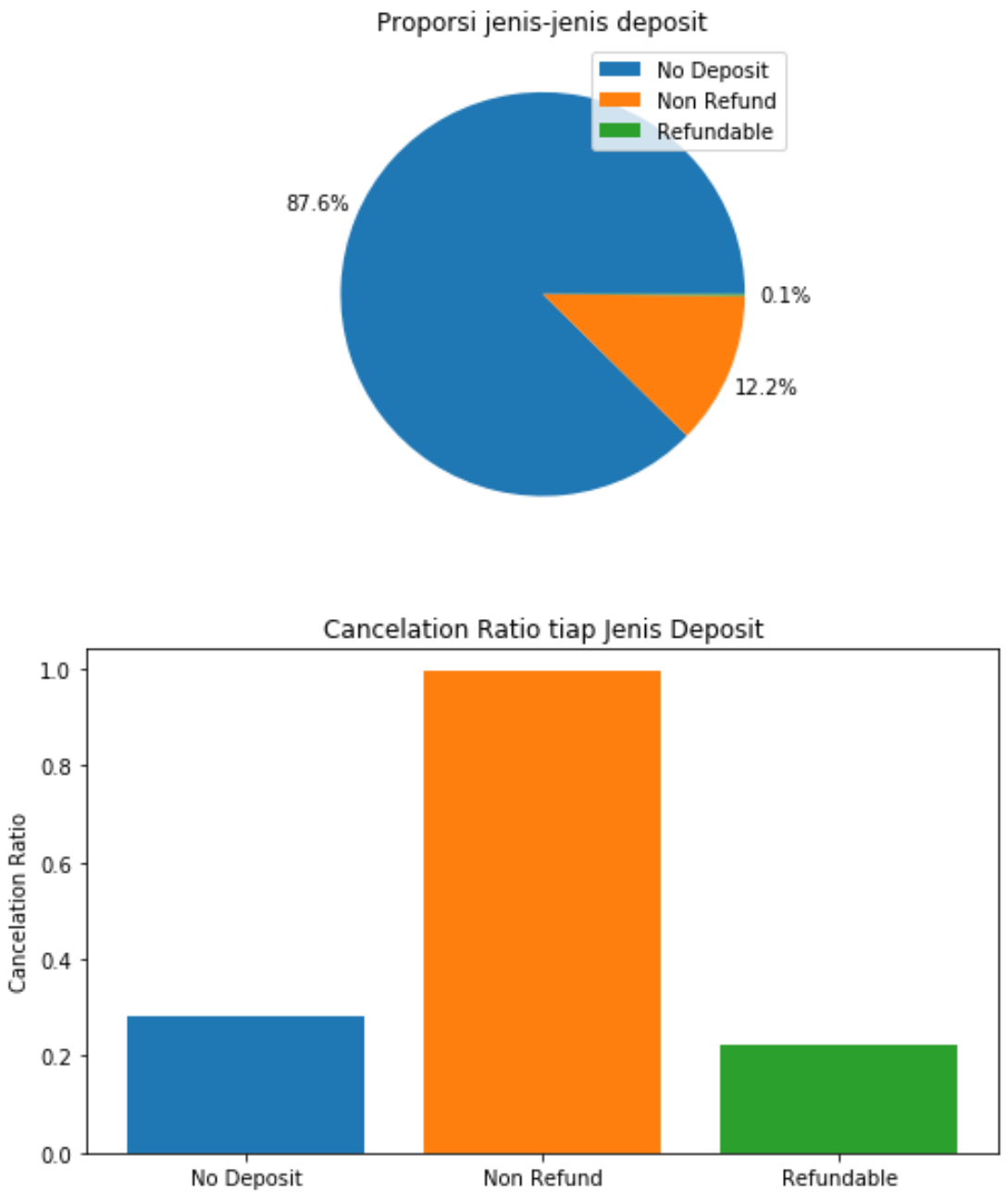
Berdasarkan paparan informasi di atas, pihak manajemen hotel dapat membuat kebijakan seperti **memperketat proses pemesanan untuk waktu kedatangan yang jatuh di kisaran pertengahan tahun** dan sebaliknya lebih melonggarkan pemesanan untuk awal dan akhir tahun. Selain itu, **dana *marketing* ataupun dana pemberian insentif kepada konsumen bisa lebih difokuskan di awal dan akhir tahun** untuk meningkatkan *revenue* pemesanan kamar secara keseluruhan.

5.4. Analisis Fitur 'Deposit_type' terhadap Keberhasilan Pemetaan Pembatalan Pesanan Kamar Hotel

Poin analisis yang lain adalah peninjauan fitur jenis deposit atau 'deposit_type'. Sesuai definisi yang telah disampaikan pada Tabel 3 feature dictionary, terdapat 3 (tiga) jenis sistem deposit yang digunakan oleh pihak manajemen hotel, yaitu: *no deposit, non refund,* dan *refundable*. Dalam rangka memahami karakteristik fitur 'deposit_type' dengan lebih baik, dilakukan visualisasi data khususnya perihal proporsi masing-masing jenis deposit dan perbandingan rasio pembatalan untuk masing-masing jenis deposit. Berikut merupakan *syntax* yang digunakan untuk melakukan visualisasi terkait:

```
fig = plt.figure(figsize=(8,10))
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)

ax1.pie(data['deposit_type'].value_counts(),
autopct='%1.1f%%', pctdistance=1.2)
ax1.legend(data['deposit_type'].value_counts().index)
```

```
ax1.set_title("Proporsi jenis-jenis deposit")

ax2.bar(data.groupby('deposit_type')['is_canceled'].mean().ind
ex, data.groupby('deposit_type')['is_canceled'].mean(),
color=['#1F77B4', '#FF7F0E', '#2CA02C'])
ax2.set_ylabel("Cancelation Ratio")
ax2.set_title("Cancelation Ratio tiap Jenis Deposit")
```



Figur 15 hasil visualisasi jenis deposit

Hasil visualisasi pada Figur 15 menunjukkan fenomena tak biasa yang menarik untuk dibahas. Pada hasi visualisasi, terlihat bahwa rasio pembatalan untuk tipe deposit *non refund* sangatlah tinggi, bahkan hingga mendekati 100%. Hal ini kemungkinan besar disebabkan oleh

manajemen hotel yang sudah memberlakukan sistem *non refund* pada pesanan-pesanan yang sedari awal sudah diduga untuk dibatalkan.

Berdasarkan informasi yang diperoleh tersebut, dapat diberikan masukkan kepada pihak manajemen untuk tetap **memberlakukan sistem *non refund* untuk pesanan-pesanan yang berkemungkinan besar akan dibatalkan**. Jika dikaitkan dengan topik bahasan analisis yang sebelumnya telah dipaparkan, maka pihak manajemen dapat **memberlakukan sistem *non refund* pada minggu-minggu sibuk seperti yang berada pada pertengahan tahun**. Selain itu, untuk mengakomodasi kerugian akibat pembatalan pesanan secara umum, bisa juga diberlakukan **sistem pembayaran *non refund* pada beberapa hari sebelum waktu kedatangan**.

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan

1. Telah berhasil dibangun suatu model prediksi untuk mengklasifikasikan pesanan yang berpeluang untuk dibatalkan dan pesanan yang kurang berpeluang untuk dibatalkan, dengan akurasi 83,56%. Model yang dibangun berbasis pada model *random forest* dengan pengaturan *hyperparameter* 'max_features' sebesar 10 dan 'n_estimators' sebesar 200. Proses pemodelan menggunakan berbagai macam fitur pilihan yang diambil secara langsung maupun fitur yang direkayasa melalui *feature engineering*.
2. Telah berhasil diperoleh 4 (empat) faktor-faktor penting yang berperan aktif dalam menentukan keberhasilan suatu pesanan. Keempat faktor tersebut diperoleh berdasarkan *feature importance* atau rata-rata *gini score* terbesar yang diperoleh pada proses pemodelan. Keempat fitur tersebut diantaranya: *lead time,* ADR, *arrival date week number,* dan *deposit type*.

6.2. Saran

1. Sehubungan dengan *lead time,* terdapat 3 (tiga) usulan yaitu:
    a. Memberlakukan sistem maksimal *lead time* pemesanan
    b. Memberikan insentif langsung (insentif yang diberikan di awal pemesanan, misal potongan harga) untuk pesanan dengan waktu *lead time* yang rendah untuk menggeser pola pemesanan dari pemesanan jangka panjang (*lead time* tinggi) yang besar peluang untuk dibatalkan, menjadi pemesanan jangka pendek (*lead time* rendah) yang lebih kecil peluang untuk dibatalkannya.
    c. Memberikan insentif tidak langsung (insentif yang diberikan di akhir proses pemesanan, misal *cashback* ataupun kupon diskon untuk pemesanan berikutnya) untuk pesanan dengan waktu *lead time* jangka panjang.
2. Sehubungan dengan ADR atau *average daily rate*, terdapat usulan berupa memberikan insentif untuk pemesanan kamar hotel yang menghasilkan *revenue* lebih tinggi pada kondisi ADR rendah untuk meningkatkan ADR, serta mengalokasikan biaya insentif untuk meningkatkan aspek pelayanan lain apabila ADR sudah cukup tinggi. Kondisi ADR rendah adalah ketika nilai ADR berada di bawah 50 (lima puluh), sementara kondisi ADR tinggi adalah ketika nilai ADR berada di atas 150 (seratus lima puluh).
3. Sehubungan dengan *arrival date week number*, terdapat usulan berupa:
    a. Memperketat proses pemesanan untuk waktu kedatangan yang jatuh di kisaran pertengahan tahun yang berpeluang lebih besar untuk dibatalkan dibandingkan dengan pemesanan untuk awal dan akhir tahun.
    b. Melonggarkan pemesanan dan mengalokasikan dana insentif pada masa-masa awal dan akhir tahun untuk menarik minat konsumen & meningkatkan *revenue*.

4. Sehubungan dengan *deposit type*, terdapat usulan berupa:
    a. Memberlakukan sistem *non refund* untuk pesanan-pesanan yang berkemungkinan besar akan dibatalkan.
    b. Lebih memfokuskan sistem *non refund* pada minggu-minggu sibuk dan rawan seperti yang berada pada pertengahan tahun.
    c. Memberlakukan sistem pembayaran *non refund* pada beberapa hari sebelum waktu kedatangan untuk mengakomodasi kerugian akibat pembatalan pesanan secara mendadak.

DAFTAR PUSTAKA

Antonio, N., Almeida, A. d., & Nunes, L. (2019). Hotel booking demand datasets. *Data in Brief*, 41-49.

Brownlee, J. (2020, August 15). *Discover Feature Engineering, How to Engineer Features and How to Get Good at It*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

catboost. (2021, May 28). *Overview of CatBoost.* Retrieved from catboost.ai: https://catboost.ai/docs/concepts/about.html

Dorogush, A. V., Ershov, V., & Yandex, A. G. (2018). *CatBoost: gradient boosting with categorical features support.* arXiv.

Eurostat. (2021, May 5). *Tourism statistics - characteristics of tourism trips*. Retrieved from Eurostat: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Tourism_statistics_-_characteristics_of_tourism_trips

Freed, J. (2016, August 12). *Cancellation trends cause headaches for hotels*. Retrieved from duettocloud.com: https://www.duettocloud.com/library/cancellation-trends-cause-headaches-hotels

Hargrave, M. (2020, October 27). *Average Daily Rate (ADR)*. Retrieved from Investopedia: https://www.investopedia.com/terms/a/average-daily-rate.asp

Himmetoglu, B. (2017, February). *Stacking Models for Improved Predictions*. Retrieved from KDnuggets: https://www.kdnuggets.com/2017/02/stacking-models-imropved-predictions.html

IBM. (2021, May 29). *Introduction to Missing Values*. Retrieved from ibm: https://www.ibm.com/docs/en/spss-statistics/27.0.0?topic=values-introduction-missing

Krishnan, V., Mann, R., Seitzman, N., & Wittkamp, N. (2020, June 10). *Hospitality and COVID-19: How long until 'no vacancy' for US hotels?* Retrieved from McKinsey: https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/hospitality-and-covid-19-how-long-until-no-vacancy-for-us-hotels

Liaw, A., & Wiener, M. (2001). Classification and Regression by RandomForest.

Pandey, P. (2019, November 25). *Data Preprocessing: Concept*. Retrieved from towardsdatascience: https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825

Paul, S. (2018, August 15). *Hyperparameter Optimization in Machine Learning Models*. Retrieved from datacamp: https://www.datacamp.com/community/tutorials/parameter-optimization-machine-learning-models?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupi

d=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=3326

readthedocs. (2021, May 28). *XGBoost Documentation.* Retrieved from readthedocs.io: https://xgboost.readthedocs.io/en/latest/index.html

Scikit Learn. (2021, May 29). *sklearn.ensemble.RandomForestClassifier*. Retrieved from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Scikit Learn. (2021, May 29). *sklearn.linear_model.LogisticRegression*. Retrieved from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Shaikh, R. (2018, October 28). *Feature Selection Techniques in Machine Learning with Python*. Retrieved from towardsdatascience: https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:~:text=Feature%20Selection%20is%20the%20process,learn%20based%20on%20irrelevant%20features.