**Kopekan UTS Feature Engineering Open-Book**

***By Lathif Ramadhan (5231811022), Bantul Regency, Special Region of Yogyakarta***

***Data Science Departement, University of Technology Yogyakarta***

**Over Engineering vs Under Engineering vs Middle Engineering**

**Over-Engineering**

**Over-Engineering** adalah kondisi di mana kita menambahkan terlalu banyak fitur atau melakukan terlalu banyak transformasi yang sebenarnya tidak diperlukan pada data. Dalam konteks *feature engineering*, ini berarti kita mungkin menambahkan kompleksitas ekstra ke dataset kita, baik melalui pembuatan fitur yang sangat spesifik maupun melakukan terlalu banyak transformasi, sehingga hasilnya justru dapat memperburuk performa model.

**Karakteristik Over-Engineering**

1. **Terlalu Banyak Fitur Kompleks**: Fitur-fitur yang ditambahkan mungkin terlalu spesifik atau terlalu kompleks sehingga sebenarnya hanya berguna dalam kondisi yang sangat terbatas. Contohnya, menambahkan fitur seperti "umur kali tinggi badan dibagi dengan jumlah hari dalam setahun" bisa jadi tidak relevan dan tidak berkontribusi pada performa model.
2. **Transformasi Berlebihan**: Melakukan terlalu banyak teknik transformasi, seperti mengubah setiap fitur kategorikal menjadi *polynomial features* atau menerapkan berbagai jenis skala secara bersamaan (misalnya normalisasi, standarisasi, dan binning) pada dataset yang sama.
3. **Susah Ditafsirkan**: Fitur yang terlalu kompleks atau tidak langsung memiliki makna yang sulit dipahami oleh model maupun pengguna. Ini juga bisa menyulitkan proses interpretasi hasil analisis

**Mengapa Over-Engineering Tidak Diinginkan?**

1. **Overfitting**: Salah satu kelemahan utama dari *over-engineering* adalah risiko *overfitting*, di mana model menjadi terlalu cocok dengan data pelatihan dan kehilangan kemampuan untuk menggeneralisasi terhadap data baru. Fitur yang terlalu spesifik atau rumit mungkin hanya relevan dalam dataset yang digunakan saat ini, dan tidak cocok untuk data di luar sampel.
2. **Membebani Proses Komputasi**: Menambahkan fitur dan transformasi berlebihan bisa memperlambat proses komputasi. Setiap fitur tambahan yang tidak relevan membutuhkan lebih banyak sumber daya untuk pemrosesan, yang mungkin tidak proporsional dengan peningkatan performa.
3. **Penurunan Interpretabilitas**: Model yang menggunakan banyak fitur atau fitur yang terlalu kompleks akan sulit dijelaskan atau dipahami oleh pengguna. Ini khususnya masalah dalam aplikasi di mana interpretasi sangat penting, seperti model prediktif untuk keputusan bisnis atau medis.

**Kapan Over-Engineering Bisa Berguna?**

Ada beberapa situasi di mana penambahan fitur kompleks atau transformasi berlebihan mungkin membantu, terutama dalam kasus model yang sangat kompleks seperti jaringan saraf dalam (*deep learning*) yang memiliki kapasitas besar untuk memahami kompleksitas data. Namun, dalam kebanyakan kasus dan terutama dalam *machine learning* tradisional, over-engineering lebih sering membawa dampak negatif.

**Keunggulan Over-Engineering (Ketika Diterapkan dengan Hati-hati)**

1. **Model yang Sangat Spesifik**: Dalam beberapa kasus yang sangat spesifik, *over-engineering* bisa membantu menciptakan model yang tepat guna untuk kebutuhan khusus. Misalnya, dalam pemodelan untuk aplikasi medis, penambahan fitur spesifik yang diperoleh dari *domain knowledge* bisa membantu.
2. **Eksplorasi Data yang Mendalam**: Melalui over-engineering, kita seringkali bisa menemukan pola atau anomali yang tidak terdeteksi dalam analisis awal. Ini bisa membantu dalam tahap awal eksplorasi data ketika tujuan utamanya adalah memahami kompleksitas dataset.

**Kesimpulan**

*Over-engineering* pada intinya adalah tentang keseimbangan antara eksplorasi dan relevansi fitur. Meskipun menambahkan fitur yang lebih rumit bisa membuka potensi penemuan baru, terlalu banyak fitur justru menimbulkan efek negatif yang mungkin lebih besar dari manfaatnya. Sehingga, over-engineering lebih cocok dipakai dengan sangat hati-hati dan lebih sering dihindari dalam *pipeline* feature engineering.

**Under-Engineering**

**Under-Engineering** adalah kondisi di mana kita kurang memberikan perhatian atau kurang melakukan pengolahan terhadap data fitur yang digunakan dalam model. Artinya, dataset yang dipakai belum diproses atau diperkaya dengan cukup sehingga belum memberikan informasi yang optimal kepada model, sehingga performanya pun kurang maksimal.

**Karakteristik Under-Engineering**

1. **Kekurangan Fitur yang Relevan**: Dataset yang digunakan mungkin hanya memiliki fitur-fitur dasar tanpa ada upaya untuk menambah fitur yang lebih representatif atau berguna untuk membantu model memahami hubungan dalam data.
2. **Minimnya Transformasi**: Dalam kasus under-engineering, fitur-fitur numerik atau kategorikal mungkin tidak diubah atau diolah, seperti normalisasi pada fitur numerik atau encoding pada fitur kategorikal. Akibatnya, data yang dipakai dalam model tidak dalam format yang paling optimal untuk proses pembelajaran.
3. **Mengabaikan Domain Knowledge**: Salah satu kekurangan dari under-engineering adalah ketika pengetahuan mendalam dari domain tertentu (seperti ekonomi, medis, atau sosial) tidak diintegrasikan dalam bentuk fitur baru yang bisa memberi nilai tambah.

**Mengapa Under-Engineering Bisa Menghambat Performasi Model?**

1. **Kurang Optimalnya Kinerja Model**: Tanpa fitur atau transformasi yang memadai, model mungkin kesulitan untuk mengidentifikasi pola atau hubungan dalam data. Akibatnya, akurasi model atau performanya pada metrik lain menjadi lebih rendah dibandingkan dengan model yang menggunakan fitur-fitur yang lebih relevan.
2. **Rentan Terhadap Kesalahan Interpretasi**: Karena data belum diproses atau diringkas dengan baik, model dapat membuat prediksi atau kesimpulan yang tidak akurat. Fitur yang kurang representatif dapat menyebabkan model salah memahami pola, terutama dalam situasi di mana perbedaan kecil pada fitur bisa membuat perubahan besar dalam prediksi.
3. **Kurangnya Generalisasi**: Model yang dibangun dengan fitur yang minimal dan tanpa transformasi cenderung kurang mampu beradaptasi dengan data baru. Karena pola dalam data mungkin tidak terlihat atau tidak terwakili dengan baik, model menjadi kurang akurat ketika diuji pada dataset yang berbeda.

**Kapan Under-Engineering Dapat Dianggap Menguntungkan?**

Meskipun under-engineering seringkali dianggap sebagai kekurangan, ada beberapa kasus tertentu di mana pendekatan ini mungkin memiliki keuntungan, terutama dalam skenario di mana:

* **Prototyping atau Eksperimen Awal**: Ketika hanya ingin mencoba atau memvalidasi model dengan cepat tanpa terlalu memproses data, under-engineering bisa menjadi langkah awal yang baik.
* **Simplicity Over Complexity**: Dalam beberapa kasus, menjaga data sesederhana mungkin bisa memudahkan proses interpretasi hasil, terutama jika fitur atau transformasi yang kompleks tidak menambah banyak nilai pada pemahaman kita terhadap data.

**Keunggulan Under-Engineering (Ketika Digunakan Tepat)**

1. **Efisiensi Waktu dan Sumber Daya**: Under-engineering membutuhkan waktu dan sumber daya yang lebih sedikit karena tidak ada banyak proses tambahan untuk menambahkan atau mentransformasi fitur. Hal ini memungkinkan pengembangan model yang lebih cepat dan lebih mudah.
2. **Interpretasi Model yang Mudah**: Model yang dibangun dengan data yang minimal dan fitur yang sederhana akan lebih mudah diinterpretasi. Ini sangat bermanfaat dalam konteks di mana penjelasan prediksi sangat penting, misalnya dalam model prediktif sederhana atau pengujian konsep.

**Kesimpulan**

Under-engineering dapat menjadi masalah yang serius jika fitur dalam dataset tidak cukup untuk menggambarkan pola yang ada, sehingga model kehilangan informasi yang diperlukan untuk performa yang optimal. Namun, dalam beberapa kasus tertentu, under-engineering dapat berguna, terutama ketika tujuan utama adalah kesederhanaan, efisiensi, atau ketika berada dalam tahap awal eksplorasi data.

**Middle Engineering**

**Middle Engineering** adalah pendekatan yang seimbang dalam feature engineering. Di sini, kita menambahkan cukup fitur dan melakukan transformasi yang diperlukan untuk meningkatkan performa model tanpa menambah kompleksitas yang berlebihan atau mengabaikan aspek penting dari data. *Middle engineering* bertujuan untuk mencapai titik tengah yang optimal, di mana fitur yang digunakan cukup kaya dalam informasi namun tetap relevan, tanpa membebani komputasi atau menambah risiko *overfitting*.

**Karakteristik Middle Engineering**

1. **Fitur yang Relevan dan Optimal**: Dalam pendekatan middle engineering, fitur yang ditambahkan benar-benar dipilih berdasarkan relevansi dan pengaruh positif terhadap model. Tidak terlalu banyak fitur yang kompleks, namun cukup untuk membantu model mengenali pola dalam data dengan baik.
2. **Transformasi yang Tepat Guna**: Middle engineering melibatkan transformasi data yang benar-benar diperlukan untuk memperbaiki format data agar mudah dipahami oleh model, seperti normalisasi atau encoding untuk fitur kategorikal. Transformasi dilakukan secara proporsional, tanpa berlebihan.
3. **Integrasi Domain Knowledge dengan Efisien**: Middle engineering mengambil aspek-aspek kunci dari *domain knowledge* yang berguna untuk analisis data. Fitur-fitur yang ditambahkan berasal dari pemahaman mendalam mengenai data dan relevan dalam konteks aplikasi, namun tidak menambah kompleksitas yang berlebihan.

**Keuntungan dari Middle Engineering**

1. **Optimalisasi Kinerja Model**: Karena middle engineering menambahkan fitur yang relevan dan menerapkan transformasi yang diperlukan saja, model menjadi lebih mudah dalam memahami data sehingga kinerjanya menjadi optimal. Pendekatan ini meningkatkan *accuracy* atau *precision* model tanpa meningkatkan risiko overfitting atau underfitting.
2. **Efisiensi dalam Komputasi**: Middle engineering menggunakan fitur yang sudah dipilih dan transformasi yang proporsional, sehingga tidak membebani proses komputasi. Model yang dihasilkan tetap ringan dan cepat dalam pemrosesan, namun tetap mampu melakukan prediksi dengan akurat.
3. **Interpretabilitas yang Baik**: Dengan fitur-fitur yang relevan namun tidak kompleks, model menjadi lebih mudah diinterpretasi. Pendekatan ini memberikan keuntungan dalam konteks bisnis atau penelitian di mana interpretasi hasil sangat penting.
4. **Generalisasi yang Lebih Baik**: Karena middle engineering menghindari fitur yang berlebihan, model yang dihasilkan lebih siap untuk data baru. Ini membantu model menggeneralisasi dengan baik, sehingga prediksinya tetap akurat meskipun terdapat variasi pada dataset yang berbeda.

**Tantangan dalam Middle Engineering**

1. **Penentuan Keseimbangan yang Tepat**: Mencapai keseimbangan yang ideal dalam middle engineering bisa menjadi tantangan, terutama dalam menentukan fitur dan transformasi yang tepat tanpa menambah atau mengurangi informasi yang berguna.
2. **Adaptasi terhadap Kompleksitas Data**: Pada dataset yang sangat kompleks, menemukan fitur yang paling informatif dan transformasi yang optimal bisa memerlukan eksperimen berulang dan evaluasi kinerja model secara mendetail.

**Kesimpulan**

*Middle Engineering* adalah pendekatan terbaik dalam feature engineering karena memberikan keseimbangan antara kompleksitas dan kesederhanaan fitur. Dengan menerapkan middle engineering, kita bisa mendapatkan model yang efisien, akurat, dan tetap mudah diinterpretasi. Pendekatan ini biasanya menjadi tujuan ideal dalam *machine learning* karena mampu memanfaatkan kekuatan dari kedua sisi over-engineering dan under-engineering tanpa terkena kelemahan utama dari masing-masing pendekatan tersebut.

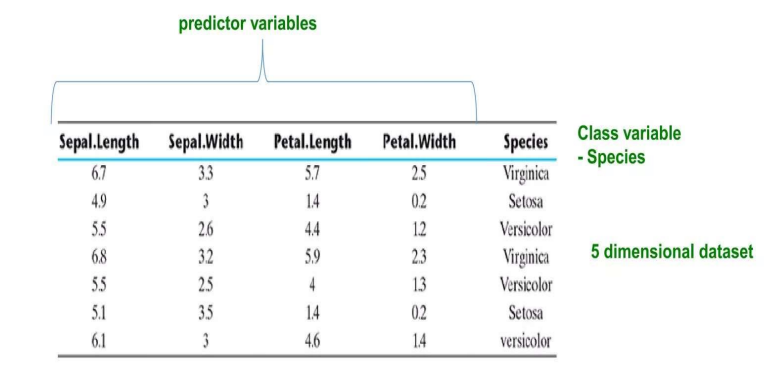
**PPT UTS**

**Dasar-dasar Feature Engineering (FE)**

* proses menerjemahkan kumpulan data menjadi fitur sehingga fitur tersebut dapat mewakili kumpulan data secara lebih efektif dan menghasilkan kinerja pembelajaran yang lebih baik.
* mengacu pada manipulasi penambahan, penghapusan, kombinasi, mutasi set data Anda untuk meningkatkan pelatihan model pembelajaran mesin, yang menghasilkan kinerja yang lebih baik dan akurasi yang lebih tinggi.
* bagian dari **kegiatan persiapan/preparatory analysis**
* Ia bertanggung jawab untuk mengambil **data masukan mentah** dan **mengubahnya** menjadi **fitur-fitur yang selaras** dan **siap digunakan** oleh **model pembelajaran mesin**.
* FE merangkum berbagai teknik rekayasa data seperti memilih fitur yang relevan, menangani data yang hilang, mengkodekan data, dan menormalkannya.
* **Memiliki dua elemen utama**
  + transformasi fitur / *feature transformation*
  + pemilihan subset fitur / *feature subset selection*

**Feature (Hal 3)**

* Fitur adalah **atribut kumpulan data** yang digunakan dalam proses pembelajaran mesin.
* **Pemilihan subset fitur** yang bermakna bagi pembelajaran mesin merupakan **sub-area rekayasa fitur**.
* Fitur-fitur dalam kumpulan data juga disebut **dimensinya**.
* himpunan data yang memiliki **fitur 'n'/*’n’ features*** disebut **himpunan data n-dimensi/*n-dimensional dataset***.



**Feature Transformation (Hal 4)**

* Terapkan rumus matematika ke kolom (fitur) tertentu dan ubah nilai yang berguna untuk analisis selanjutnya.
* Membuat fitur-fitur baru dari fitur-fitur yang sudah ada yang dapat membantu dalam meningkatkan kinerja model.
* Seperangkat fitur (m) untuk membuat himpunan fitur baru (n) sambil mempertahankan informasi sebanyak mungkin. Ia memiliki dua elemen utama:

1. konstruksi fitur 2. ekstraksi fitur Keduanya terkadang dikenal sebagai penemuan fitur

**Ada dua tujuan berbeda dari transformasi fitur:**

1. Mencapai rekonstruksi terbaik dari fitur asli dalam kumpulan data

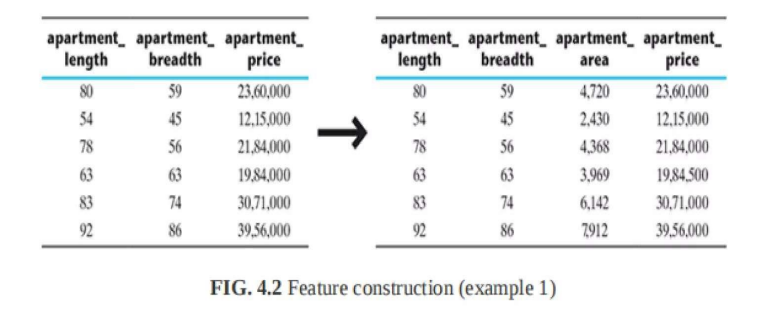
2. Mencapai efisiensi tertinggi dalam tugas pembelajaran

**Feature Construction (Hal 5)**

Melibatkan transformasi serangkaian fitur masukan yang diberikan untuk menghasilkan serangkaian fitur baru yang lebih canggih. Kumpulan data ini memiliki tiga fitur -

* *panjang apartemen,*
* *lebar apartemen, dan*
* *harga apartemen.*

Jika digunakan sebagai masukan untuk masalah regresi, data tersebut dapat menjadi data pelatihan untuk model regresi. Jadi dengan data pelatihannya, model tersebut seharusnya dapat memprediksi harga sebuah apartemen yang harganya belum diketahui atau yang baru saja dijual. Namun, daripada menggunakan **panjang dan lebar** apartemen sebagai prediktor, jauh lebih nyaman dan lebih masuk akal untuk menggunakan area apartemen, yang bukan merupakan fitur yang ada dalam kumpulan data. Maka fitur tersebut, yaitu **luas apartemen**, dapat ditambahkan pada kumpulan data tersebut. Dengan kata lain, kami mengubah kumpulan data tiga dimensi menjadi kumpulan data empat dimensi, dengan **area apartemen** fitur yang baru 'ditemukan' ditambahkan ke kumpulan data asli.



* ketika fitur memiliki nilai kategoris dan pembelajaran mesin membutuhkan input nilai numerik
* ketika fitur memiliki nilai numerik (kontinu) dan perlu diubah menjadi nilai ordinal
* ketika konstruksi fitur khusus teks perlu dilakukan

Data ordinal adalah bilangan bulat diskrit yang dapat diperingkat atau diurutkan

**Encoding Categorical (Nominal) Variables**

**A white rectangular box with black text

Description automatically generated**

A white rectangular box with black text

Description automatically generated

A screen shot of a computer

Description automatically generated

**Encoding Categorical (Ordinal) Variables**

A table with text and numbers

Description automatically generated with medium confidence

* Nilai adalah variabel ordinal dengan nilai A, B, C, dan D.
* Untuk mengubah variabel ini menjadi variabel numerik, kita dapat membuat fitur num\_grade yang memetakan nilai numerik terhadap setiap nilai ordinal.
* dipetakan ke nilai 1, 2, 3, dan 4 dalam variabel yang ditransformasikan

**Transforming Numeric (continuous) Features to Categorical Value**

* Sebagai prediksi kategori harga real estat, yang merupakan masalah klasifikasi.
* Dalam kasus tersebut, kita dapat mengelompokkan data numerik ke dalam beberapa kategori berdasarkan rentang data. Dalam konteks harga real estat contoh prediksi, kumpulan data asli memiliki fitur numerik apartment\_price

A table of numbers and a few numbers

Description automatically generated with medium confidence

A table of numbers and text

Description automatically generated with medium confidence

**Text Spesific Feature Extraction**

Teks boleh dibilang merupakan media komunikasi yang paling dominan. Penambangan teks merupakan bidang penelitian yang pentingsifat data yang tidak terstruktur. Semua model pembelajaran mesin memerlukan data numerik sebagai input. Jadi data teks dalam kumpulan data perlu diubah menjadi fitur numerik. Contoh:

* Facebook atau saluran mikro-blog seperti Twitter atau email atau layanan pesan singkat seperti
* Whatsapp, Teks memainkan peran utama dalam arus informasi.

**Vektorisasi/Vectorization:**

* Mengubah teks menjadi vektor/array
* Untuk mengubah teks menjadi vektor integer (atau boolean, atau bilangan mengambang).
* vektor adalah daftar dengan n posisi.

A white background with red text

Description automatically generated

A screenshot of a data analysis

Description automatically generated

**Tiga langkah utama/Three Major Step**

**1. Tokenisasi:** Untuk membuat token pada **korpus(corpus)**, spasi kosong dan tanda baca digunakan sebagai pembatas untuk memisahkan mengeluarkan kata-kata, atau token. **Korpus** adalah kumpulan teks atau audio autentik yang diorganisasikan menjadi kumpulan data.

**2. Hitung:** Kemudian jumlah kemunculan setiap token dihitung, untuk setiap dokumen.

**3. Normalisasi:** Token diberi bobot dengan mengurangi kepentingan saat muncul di sebagian besar dokumen. Suatu matriks kemudian dibentuk dengan tiap token mewakili suatu kolom, dan dokumen tertentu dalam korpus mewakili tiap baris. Setiap sel berisi jumlah kemunculan token dalam dokumen tertentu.

Matriks ini dikenal sebagai matriks istilah **dokumen / matriks istilah-dokumen---🡪 as a document-term matrix/term-document matrix**



A screen shot of a chart

Description automatically generated

**Ekstraksi Fitur/Feature Extraction**

Fitur-fitur baru dibuat dari kombinasi fitur-fitur asli. Operator untuk menggabungkan fitur asli meliputi:

**1. Untuk fitur Boolean:** Konjungsi, Disjungsi, Negasi, dll.

**2. Untuk fitur nominal:** Produk Cartesian, M dari N, dll.

**3. Untuk fitur numerik:** Min, Maks, Penjumlahan, Pengurangan, Perkalian, Pembagian, Rata-rata, Kesetaraan, Ketimpangan, dsb.



A white paper with black text and numbers

Description automatically generated

A black text on a white background

Description automatically generated

**Contoh Soal dan Jawaban beserta kasusnya**

**Kasus UTS: Penerapan Feature Engineering pada Data Penjualan Online**

**Deskripsi Kasus:**

Anda adalah seorang data scientist yang bekerja untuk sebuah perusahaan e-commerce. Perusahaan ini ingin menganalisis data penjualan mereka untuk meningkatkan performa bisnis. Anda diberikan sebuah dataset yang mencakup informasi penjualan dari beberapa bulan terakhir, yang berisi fitur-fitur berikut:

1. **Product Category** (kategori produk) – Kategorikal (e.g., Elektronik, Pakaian, Makanan)
2. **Price** (harga produk) – Numerik
3. **Sales Quantity** (jumlah penjualan) – Numerik
4. **Discount** (diskon yang diterima) – Numerik (dalam persen)
5. **Customer Age** (usia pelanggan) – Numerik
6. **Customer Region** (wilayah pelanggan) – Kategorikal (e.g., Jakarta, Bandung, Surabaya)
7. **Purchase Date** (tanggal pembelian) – Tanggal

Tugas Anda adalah untuk menerapkan berbagai teknik **feature engineering** untuk memproses data ini agar model machine learning dapat menggunakannya dengan lebih efektif. Anda diminta untuk melakukan beberapa langkah berikut:

1. **Feature Label dan Variabel Label**: Tentukan fitur mana yang berfungsi sebagai label (target) dan mana yang merupakan variabel fitur (independent variables).
2. **Feature Transformation**: Lakukan transformasi data dengan normalisasi dan encoding pada fitur-fitur numerik dan kategorikal yang relevan.
3. **Feature Construction dan Extraction**: Bangun fitur baru yang dapat membantu meningkatkan performa model. Misalnya, ekstraksi informasi dari tanggal pembelian.
4. **Data Extraction dan Data Construction**: Ambil subset data yang relevan dan konstruksi fitur baru yang dapat mengoptimalkan model.

**Soal-soal dan Jawabannya**

**Soal 1: Feature Label dan Variabel Label**

**Soal:** Dalam kasus penjualan online di atas, tentukan fitur mana yang harus dijadikan **label (target)** dan fitur mana yang merupakan **variabel label (features)** yang digunakan untuk memprediksi target tersebut. **Jawaban:**

* **Label (Target)**: Fitur yang menjadi target prediksi adalah **Sales Quantity** karena kita ingin memprediksi jumlah penjualan berdasarkan faktor-faktor lainnya.
* **Variabel Fitur (Features)**: Fitur-fitur yang digunakan untuk memprediksi **Sales Quantity** adalah **Product Category**, **Price**, **Discount**, **Customer Age**, **Customer Region**, dan **Purchase Date**.

**Soal 2: Normalisasi Fitur Numerik**

**Soal:** Jelaskan langkah-langkah yang perlu dilakukan untuk melakukan normalisasi pada fitur numerik seperti **Price**, **Sales Quantity**, dan **Discount**. **Jawaban:**

1. **Identifikasi fitur numerik**: Fitur numerik yang perlu dinormalisasi adalah **Price**, **Sales Quantity**, dan **Discount**.
2. **Pilih teknik normalisasi**: Teknik yang paling umum digunakan adalah **Min-Max Normalization**.
3. **Normalisasi**: Terapkan normalisasi dengan formula:

A black background with white text

Description automatically generated

Di mana XXX adalah nilai asli fitur, min⁡(X)\min(X)min(X) adalah nilai minimum dalam kolom, dan max⁡(X)\max(X)max(X) adalah nilai maksimum dalam kolom.

1. **Penerapan**: Dengan menggunakan pustaka Python seperti scikit-learn, kita bisa menggunakan MinMaxScaler() untuk menormalisasi ketiga fitur ini sehingga nilai mereka berada dalam rentang [0, 1].

**Soal 3: One-Hot Encoding untuk Fitur Kategorikal**

**Soal:** Bagaimana cara melakukan **one-hot encoding** untuk fitur **Product Category** dan **Customer Region**? **Jawaban:**

1. **Identifikasi fitur kategorikal**: Fitur yang perlu di-encode adalah **Product Category** dan **Customer Region**.
2. **Penerapan One-Hot Encoding**:
   * **Product Category**: Kategori produk seperti **Elektronik**, **Pakaian**, dan **Makanan** akan dipisah menjadi tiga kolom:
     + Elektronik: 1 atau 0
     + Pakaian: 1 atau 0
     + Makanan: 1 atau 0
   * **Customer Region**: Kolom ini akan dibagi menjadi tiga kolom:
     + Jakarta: 1 atau 0
     + Bandung: 1 atau 0
     + Surabaya: 1 atau 0
3. **Fungsi**: Fungsi one-hot encoding ini akan mengubah data kategorikal menjadi format numerik biner sehingga model dapat menggunakannya.

**Soal 4: Feature Construction dan Extraction**

**Soal:** Dari fitur **Purchase Date**, buatlah fitur baru yang dapat membantu meningkatkan performa model. Jelaskan bagaimana caranya.

**Jawaban:**

1. **Extract Fitur Baru**: Dari **Purchase Date**, kita dapat mengekstrak beberapa informasi baru yang mungkin relevan, seperti:
   * **Month of Purchase**: Mengambil bulan dari tanggal pembelian (e.g., Januari, Februari).
   * **Day of the Week**: Mengambil informasi tentang hari dalam seminggu, karena pembelian bisa lebih banyak terjadi di akhir pekan atau hari-hari tertentu.
   * **Is Weekend**: Membuat fitur biner yang menandakan apakah pembelian terjadi pada akhir pekan (Sabtu/Minggu).
2. **Alasan**: Fitur-fitur ini dapat membantu model mempelajari pola berdasarkan waktu pembelian, misalnya pembelian lebih banyak terjadi pada akhir pekan atau bulan tertentu.

**Soal 5: Data Extraction dan Data Construction**

**Soal:** Apa yang dimaksud dengan **data extraction** dan **data construction**, dan bagaimana Anda akan mengaplikasikannya pada dataset penjualan online ini? **Jawaban:**

* **Data Extraction**: Proses ini melibatkan pemilihan subset data yang relevan berdasarkan kebutuhan analisis. Misalnya, jika kita hanya tertarik pada data penjualan dalam bulan Januari, kita akan mengekstrak data penjualan yang hanya terjadi di bulan tersebut.
* **Data Construction**: Proses ini melibatkan pembuatan fitur baru yang bisa memberikan informasi lebih dalam kepada model. Contoh dari data construction adalah membuat fitur **total sales** dengan mengalikan **Price** dengan **Sales Quantity**, atau menghitung **average discount** per kategori produk.
* **Aplikasi pada Dataset Penjualan**: Pada dataset ini, **data extraction** bisa berarti memilih subset data berdasarkan waktu atau kategori produk tertentu, sementara **data construction** bisa berarti membuat fitur baru seperti **total sales** atau **average discount per region** untuk membantu model belajar dengan lebih baik.

**Soal 6: Menangani Data Missing pada Fitur Numerik**

**Soal:** Pada dataset penjualan online, misalnya kolom **Customer Age** memiliki beberapa nilai yang hilang (missing values). Apa yang akan Anda lakukan untuk menangani masalah ini? **Jawaban:**

1. **Identifikasi Missing Values**: Pertama, kita perlu mengidentifikasi baris mana yang memiliki nilai yang hilang di kolom **Customer Age**.
2. **Strategi Pengisian Nilai Hilang**:
   * **Mean Imputation**: Mengisi nilai yang hilang dengan rata-rata dari nilai yang ada pada kolom tersebut. Ini adalah teknik yang sering digunakan ketika distribusi data tidak terlalu skewed.
   * **Median Imputation**: Jika data **Customer Age** terdistribusi skewed, lebih baik mengisi nilai yang hilang dengan nilai **median** daripada mean.
   * **Interpolation**: Dalam beberapa kasus, interpolasi dapat digunakan untuk mengisi nilai yang hilang, menggunakan informasi dari data sebelumnya atau sesudahnya.
3. **Alasan**: Memiliki data lengkap penting agar model dapat belajar dari seluruh dataset. Dengan mengisi missing values, model dapat menangani data dengan lebih baik.

**Soal 7: Menangani Outlier pada Fitur Numerik**

**Soal:** Dalam dataset penjualan online, fitur **Price** memiliki beberapa nilai yang sangat tinggi (outlier). Apa yang akan Anda lakukan untuk menangani masalah outlier ini? **Jawaban:**

1. **Identifikasi Outlier**: Outlier dapat diidentifikasi dengan menggunakan metode statistik seperti **Z-Score** (nilai lebih dari 3 atau kurang dari -3) atau **IQR (Interquartile Range)** untuk menemukan nilai yang berada jauh di luar rentang normal data.
2. **Strategi Penanganan Outlier**:
   * **Capping**: Batas nilai maksimum atau minimum untuk fitur **Price**, misalnya, jika nilai **Price** lebih dari 99 persen dari data lainnya, kita bisa menetapkan nilai tersebut ke batas tertinggi yang masih masuk akal.
   * **Log Transformation**: Untuk mengurangi dampak outlier, kita bisa menggunakan transformasi logaritma pada nilai **Price**. Ini akan mengubah distribusi data dan mengurangi pengaruh nilai ekstrem.
   * **Remove Outliers**: Jika outlier terlalu ekstrim dan tidak mewakili data yang valid, bisa dipertimbangkan untuk menghapusnya.
3. **Alasan**: Outlier yang tidak wajar dapat mengganggu model dan menyebabkan kesalahan prediksi. Mengatasi outlier dapat memperbaiki akurasi model.

**Soal 8: Feature Engineering untuk Kategori Produk**

**Soal:** Pada fitur **Product Category**, misalnya ada kategori-kategori seperti **Elektronik**, **Pakaian**, dan **Makanan**. Jika Anda ingin meningkatkan model prediksi berdasarkan kategori produk, fitur apa yang bisa Anda konstruksi atau transformasi? **Jawaban:**

1. **Feature Construction**:
   * **Total Sales per Category**: Membuat fitur yang menunjukkan **total sales** untuk setiap kategori produk. Ini dapat dihitung dengan mengalikan **Price** dan **Sales Quantity** untuk setiap kategori.
   * **Average Discount per Category**: Fitur yang menunjukkan rata-rata **Discount** yang diberikan untuk setiap kategori produk.
   * **Category Popularity**: Menghitung seberapa sering produk dalam kategori tertentu dibeli oleh pelanggan, misalnya, kategori dengan lebih banyak transaksi dapat diberi label "lebih populer".
2. **Feature Transformation**:
   * **One-Hot Encoding**: Jika kategori produk terbatas dan tidak memiliki hubungan ordinal (misalnya Elektronik, Pakaian, Makanan), **one-hot encoding** bisa sangat berguna untuk mengubah fitur ini menjadi format numerik.
   * **Target Encoding**: Jika kategori produk banyak dan memiliki hubungan dengan target (misalnya total penjualan), kita bisa menggunakan **target encoding** untuk mengganti kategori dengan rata-rata penjualan di setiap kategori.
3. **Alasan**: Dengan menambah fitur-fitur seperti total penjualan dan rata-rata diskon per kategori, model akan lebih mudah mengidentifikasi pola-pola yang relevan dalam prediksi.

**Soal 9: Feature Interaction**

**Soal:** Pada dataset penjualan online, Anda memiliki fitur **Price** dan **Discount**. Jelaskan bagaimana Anda dapat membangun fitur interaksi antara kedua fitur ini dan mengapa ini bermanfaat. **Jawaban:**

1. **Feature Interaction**:
   * **Price-Discount Interaction**: Buatlah fitur baru yang merupakan hasil perkalian antara **Price** dan **Discount**. Fitur ini dapat memberikan gambaran tentang seberapa besar pengaruh diskon terhadap harga produk. Misalnya, untuk produk dengan harga tinggi dan diskon besar, kemungkinan besar akan berdampak pada penjualan lebih besar.
   * **Discount Percentage**: Mungkin kita ingin mengetahui seberapa besar diskon yang diberikan dalam bentuk persentase terhadap harga. Ini dapat dihitung dengan rumus:

A black background with white text

Description automatically generated

* + **Benefit**: Fitur interaksi ini memberikan informasi tambahan tentang hubungan antara harga dan diskon yang mungkin memengaruhi perilaku pembelian pelanggan.

1. **Alasan**: Fitur interaksi dapat menangkap hubungan yang lebih kompleks antar fitur dan memberikan informasi yang lebih kaya bagi model, sehingga meningkatkan akurasi prediksi.

**Soal 10: Pembentukan Fitur dari Tanggal Pembelian**

**Soal:** Dari fitur **Purchase Date**, buatlah fitur tambahan yang bisa memperkaya informasi yang digunakan dalam model prediksi. Jelaskan fitur baru yang dapat dibentuk dari tanggal tersebut. **Jawaban:**

1. **Feature Extraction dari Tanggal**:
   * **Month of Purchase**: Ekstrak bulan dari **Purchase Date**. Misalnya, pembelian yang terjadi di bulan Desember mungkin memiliki pola yang berbeda dibandingkan bulan lainnya.
   * **Day of the Week**: Dapatkan hari dalam minggu (Senin, Selasa, dll.) untuk menganalisis apakah ada perbedaan dalam jumlah pembelian di akhir pekan dan hari kerja.
   * **Seasonality**: Buat fitur biner yang menandakan apakah pembelian terjadi pada musim liburan atau musim tertentu (misalnya, Natal, Tahun Baru).
   * **Is Working Day**: Tentukan apakah pembelian dilakukan pada hari kerja atau akhir pekan. Ini bisa memengaruhi pola pembelian, terutama jika pembeli lebih banyak bertransaksi saat bekerja atau selama waktu liburan.
2. **Alasan**: Dengan mengekstraksi informasi terkait waktu, model dapat mempelajari pola musiman atau perilaku pelanggan berdasarkan waktu, yang sangat berguna untuk prediksi penjualan.

**Kasus 2:**

**Kasus UTS: Penerapan Feature Engineering pada Data Pengguna Aplikasi Mobile**

**Deskripsi Kasus:**

Anda adalah seorang data scientist yang bekerja untuk sebuah perusahaan yang mengembangkan aplikasi mobile. Perusahaan ingin menganalisis data penggunaan aplikasi untuk meningkatkan retensi pengguna. Anda diberikan dataset yang berisi informasi pengguna aplikasi mobile selama 6 bulan terakhir, dengan fitur-fitur berikut:

1. **User ID** – ID unik pengguna (kategorikal)
2. **Age** – Usia pengguna (numerik)
3. **Gender** – Jenis kelamin pengguna (kategorikal: Laki-laki, Perempuan)
4. **Subscription Type** – Jenis langganan (kategorikal: Gratis, Premium)
5. **Total Time Spent in App** – Total waktu yang dihabiskan pengguna dalam aplikasi (numerik, dalam menit)
6. **Number of Sessions** – Jumlah sesi yang dilakukan pengguna dalam aplikasi (numerik)
7. **App Version** – Versi aplikasi yang digunakan (kategorikal: 1.0, 2.0, 3.0)
8. **Date of Last Login** – Tanggal login terakhir pengguna (tanggal)

Tugas Anda adalah untuk menerapkan berbagai teknik **feature engineering** pada data ini untuk mempersiapkannya bagi model machine learning yang dapat memprediksi apakah seorang pengguna akan berlangganan premium di masa mendatang.

**Soal-soal dan Jawabannya**

**Soal 1: Menentukan Feature Label dan Variabel Label**

**Soal:** Dalam kasus pengguna aplikasi mobile ini, tentukan fitur mana yang harus dijadikan **label (target)** dan fitur mana yang merupakan **variabel label (features)** untuk memprediksi target tersebut. **Jawaban:**

* **Label (Target)**: Fitur yang menjadi target prediksi adalah **Subscription Type** karena kita ingin memprediksi apakah seorang pengguna akan berlangganan versi Premium atau tidak.
* **Variabel Fitur (Features)**: Fitur-fitur yang digunakan untuk memprediksi **Subscription Type** adalah **Age**, **Gender**, **Total Time Spent in App**, **Number of Sessions**, **App Version**, dan **Date of Last Login**.

**Soal 2: Normalisasi Fitur Numerik**

**Soal:** Bagaimana cara melakukan **normalisasi** pada fitur **Total Time Spent in App** dan **Number of Sessions**? **Jawaban:**

1. **Identifikasi fitur numerik**: Fitur yang perlu dinormalisasi adalah **Total Time Spent in App** dan **Number of Sessions**.
2. **Pilih teknik normalisasi**: Teknik yang umum digunakan adalah **Min-Max Normalization**.
3. **Langkah-langkah normalisasi**:
   * Hitung nilai minimum dan maksimum dari masing-masing fitur.

Terapkan rumus Min-Max untuk setiap nilai dalam kolom:

A black background with white text

Description automatically generated

1. **Alasan**: Normalisasi membantu untuk menghindari masalah skala yang besar pada model machine learning yang sangat sensitif terhadap perbedaan skala antar fitur.

**Soal 3: One-Hot Encoding untuk Fitur Kategorikal**

**Soal:** Bagaimana cara melakukan **one-hot encoding** pada fitur **Gender**, **Subscription Type**, dan **App Version**? **Jawaban:**

1. **Identifikasi fitur kategorikal**: Fitur yang perlu di-encode adalah **Gender**, **Subscription Type**, dan **App Version**.
2. **Penerapan One-Hot Encoding**:
   * **Gender**: Kolom ini memiliki dua nilai: **Laki-laki** dan **Perempuan**. One-hot encoding akan menghasilkan dua kolom:
     + Laki-laki: 1 atau 0
     + Perempuan: 1 atau 0
   * **Subscription Type**: Kolom ini memiliki dua nilai: **Gratis** dan **Premium**. One-hot encoding akan menghasilkan dua kolom:
     + Gratis: 1 atau 0
     + Premium: 1 atau 0
   * **App Version**: Kolom ini memiliki tiga nilai: **1.0**, **2.0**, dan **3.0**. One-hot encoding akan menghasilkan tiga kolom:
     + Versi 1.0: 1 atau 0
     + Versi 2.0: 1 atau 0
     + Versi 3.0: 1 atau 0
3. **Alasan**: One-hot encoding mengubah data kategorikal menjadi format numerik sehingga model machine learning dapat memprosesnya.

**Soal 4: Feature Construction dan Extraction**

**Soal:** Dari fitur **Date of Last Login**, buatlah fitur baru yang bisa membantu meningkatkan model prediksi retensi pengguna. **Jawaban:**

1. **Extract Fitur Baru**: Dari **Date of Last Login**, kita dapat membuat fitur baru yang menggambarkan seberapa sering pengguna masuk ke aplikasi.
   * **Days Since Last Login**: Menghitung jumlah hari yang telah berlalu sejak pengguna terakhir kali login.
     + Formula:



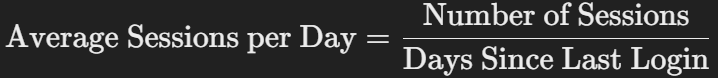
* + **Is Recently Active**: Membuat fitur biner yang menunjukkan apakah pengguna login dalam waktu 7 hari terakhir atau tidak. Jika **Days Since Last Login** <= 7, maka fitur ini bernilai 1 (aktif), jika lebih dari 7 hari, maka bernilai 0.

1. **Alasan**: Fitur **Days Since Last Login** memberikan gambaran tentang seberapa sering pengguna berinteraksi dengan aplikasi, yang merupakan faktor penting untuk memprediksi retensi dan konversi ke subscription premium.

**Soal 5: Data Extraction dan Data Construction**

**Soal:** Jelaskan apa yang dimaksud dengan **data extraction** dan **data construction**, serta bagaimana Anda akan mengaplikasikannya pada dataset pengguna aplikasi mobile ini. **Jawaban:**

* **Data Extraction**: Proses ini melibatkan pemilihan subset data yang relevan untuk analisis. Misalnya, jika kita hanya tertarik pada pengguna yang sudah berlangganan **Premium**, kita bisa mengekstrak hanya data pengguna dengan jenis langganan **Premium** untuk analisis lebih lanjut.
* **Data Construction**: Proses ini melibatkan pembuatan fitur baru yang dapat meningkatkan kemampuan model dalam memprediksi target. Misalnya, kita bisa membuat fitur **Average Sessions per Day** dengan membagi **Number of Sessions** dengan **Days Since Last Login**, memberikan gambaran seberapa aktif pengguna dalam aplikasi.
* Formula:



**Aplikasi pada Dataset**: Pada dataset ini, **data extraction** bisa berarti mengekstrak data pengguna aktif yang login lebih dari 7 hari yang lalu. Sedangkan **data construction** dapat melibatkan pembuatan fitur **Average Sessions per Day** untuk melihat interaksi pengguna dalam aplikasi.

**Soal 6: Feature Label dan Variabel Label untuk Model Prediksi Retensi**

**Soal:** Anda telah diberi tugas untuk mengembangkan model prediksi retensi pengguna aplikasi. Dari dataset yang tersedia, tentukan fitur mana yang menjadi **label** dan **variabel label** untuk prediksi **retensi pengguna**. **Jawaban:**

* **Label (Target)**: Fitur yang menjadi label adalah **Subscription Type** karena kita ingin memprediksi apakah seorang pengguna akan tetap berlangganan **Premium** atau tidak.
* **Variabel Label (Features)**: Fitur-fitur yang akan digunakan sebagai input untuk model prediksi adalah:
  + **Age**
  + **Gender**
  + **Total Time Spent in App**
  + **Number of Sessions**
  + **App Version**
  + **Days Since Last Login**

Alasan: Fitur-fitur ini akan memberikan informasi yang diperlukan untuk memahami perilaku pengguna dan memprediksi kemungkinan mereka akan tetap berlangganan premium.

**Soal 7: Menggunakan Feature Engineering untuk Mengurangi Overfitting**

**Soal:** Bagaimana Anda dapat menggunakan teknik **feature engineering** untuk mengurangi risiko **overfitting** pada model yang memprediksi apakah pengguna akan berlangganan Premium?

**Jawaban:** Untuk mengurangi risiko **overfitting**, kita dapat menggunakan beberapa teknik feature engineering berikut:

1. **Feature Selection**: Menghapus fitur yang tidak relevan atau memiliki korelasi tinggi dengan fitur lainnya. Misalnya, jika ada dua fitur yang memiliki korelasi yang sangat tinggi, memilih salah satu di antaranya dapat mengurangi kompleksitas model.
2. **Penyederhanaan Fitur**: Menggunakan teknik seperti **Principal Component Analysis (PCA)** atau **Linear Discriminant Analysis (LDA)** untuk mengurangi jumlah fitur yang digunakan dalam model. Ini akan membantu model fokus pada informasi yang lebih penting dan menghindari pembelajaran noise.
3. **Feature Transformation**: Menggunakan **log transformation** atau **scaling** untuk mereduksi distribusi yang terlalu miring pada fitur numerik. Ini akan membantu model menangani varians yang besar dengan lebih baik dan mengurangi peluang overfitting.

**Soal 8: Menggunakan Feature Interaction untuk Meningkatkan Akurasi**

**Soal:** Pada dataset pengguna aplikasi mobile, bagaimana Anda bisa menggunakan teknik **feature interaction** untuk meningkatkan akurasi model prediksi berlangganan Premium? **Jawaban:**

1. **Feature Interaction**: Untuk menangkap hubungan yang lebih kompleks antara fitur-fitur, Anda bisa menciptakan fitur interaksi antara **Total Time Spent in App** dan **Number of Sessions**.
   * **Interaction Feature**: Misalnya, kita bisa membuat fitur baru yang merupakan hasil perkalian antara **Total Time Spent in App** dan **Number of Sessions**, yaitu:

A black and white sign with white text

Description automatically generated

* + Fitur ini memberikan informasi lebih lanjut tentang berapa lama rata-rata pengguna menghabiskan waktu per sesi aplikasi, yang bisa menjadi indikator penting dalam prediksi retensi.

1. **Alasan**: Dengan menambahkan interaksi antara dua fitur ini, model akan lebih mampu memahami bagaimana interaksi antar faktor-faktor tersebut mempengaruhi keputusan pengguna untuk berlangganan premium atau tidak.

**Soal 9: Normalisasi pada Data Kategorikal**

**Soal:** Apakah normalisasi diperlukan untuk fitur **Gender**, **Subscription Type**, dan **App Version**? Jelaskan mengapa atau mengapa tidak.

**Jawaban:** **Tidak perlu melakukan normalisasi** pada fitur **Gender**, **Subscription Type**, dan **App Version**, karena:

1. **Gender** dan **Subscription Type** adalah fitur kategorikal, yang hanya merepresentasikan kategori atau label, bukan nilai numerik. Oleh karena itu, tidak ada manfaat dari normalisasi pada fitur ini.
2. **App Version** adalah fitur kategorikal dengan nilai yang terpisah (1.0, 2.0, 3.0), yang lebih baik ditangani dengan **one-hot encoding** atau **target encoding** daripada normalisasi.

Normalisasi biasanya diterapkan pada fitur numerik agar memiliki skala yang seragam, yang penting untuk model berbasis jarak atau gradient descent. Namun, fitur kategorikal tidak memerlukan normalisasi.

**Soal 10: Penanganan Missing Values pada Fitur Numerik**

**Soal:** Pada fitur **Age** dalam dataset pengguna aplikasi mobile, terdapat beberapa nilai yang hilang. Bagaimana Anda menangani missing values pada fitur **Age**? **Jawaban:**

1. **Identifikasi Missing Values**: Pertama, kita perlu memeriksa kolom **Age** untuk mengetahui di mana nilai yang hilang berada.
2. **Strategi Pengisian Missing Values**:
   * **Mean Imputation**: Jika distribusi **Age** tidak terlalu skewed, kita dapat mengisi nilai yang hilang dengan **mean** (rata-rata) dari fitur **Age**.
   * **Median Imputation**: Jika distribusi **Age** sangat skewed (misalnya banyak pengguna muda atau tua), lebih baik mengisi nilai yang hilang dengan **median** agar tidak dipengaruhi oleh nilai ekstrem.
   * **Mode Imputation (untuk data kategorikal)**: Jika ada kategori usia yang lebih sering muncul dalam data, kita bisa menggunakan mode untuk mengganti nilai yang hilang.
3. **Alasan**: Memiliki nilai yang lengkap pada fitur numerik penting untuk memungkinkan model belajar dari seluruh dataset. Menggunakan strategi imputation seperti mean atau median membantu menjaga konsistensi data dan mencegah model kehilangan informasi penting.

**Kasus 3: Prediksi Harga Sewa Apartemen Berdasarkan Karakteristik Apartemen**

Sebuah perusahaan real estate ingin mengembangkan model untuk memprediksi harga sewa apartemen berdasarkan karakteristik dari setiap unit apartemen. Dataset yang mereka kumpulkan memiliki informasi berikut untuk setiap apartemen:

1. **Apartment Length** (Panjang Apartemen) - Dalam meter.
2. **Apartment Width** (Lebar Apartemen) - Dalam meter.
3. **Apartment Location** - Lokasi apartemen yang dikategorikan sebagai "City Center", "Suburban", atau "Rural".
4. **Apartment Type** - Jenis apartemen, misalnya "Studio", "1-Bedroom", "2-Bedroom", atau "3-Bedroom".
5. **Nearby Facilities Count** (Jumlah Fasilitas Terdekat) - Jumlah fasilitas umum yang ada di dekat apartemen, seperti supermarket, sekolah, atau transportasi umum.
6. **Monthly Rent Price** (Harga Sewa Bulanan) - Dalam rupiah, ini merupakan target atau label yang ingin diprediksi oleh perusahaan.

**Soal dan Jawaban**

**Soal 1: Identifikasi Feature Label dan Variabel Label**

**Soal:** Berdasarkan dataset di atas, identifikasi feature yang menjadi **label** dan feature yang menjadi **variabel label**. Jelaskan alasan Anda. **Jawaban:**

* **Label (Target)**: **Monthly Rent Price** karena ini adalah variabel yang ingin kita prediksi.
* **Variabel Label (Features)**:
  + **Apartment Length**
  + **Apartment Width**
  + **Apartment Location**
  + **Apartment Type**
  + **Nearby Facilities Count**

Alasan: Variabel-variabel tersebut adalah karakteristik yang dapat memengaruhi harga sewa bulanan apartemen, sehingga mereka digunakan sebagai input untuk memprediksi **Monthly Rent Price**.

**Soal 2: Feature Construction untuk Menghitung Luas Apartemen**

**Soal:** Sebelum membuat model, perusahaan ingin menggunakan ukuran total luas apartemen sebagai prediktor baru. Bagaimana Anda dapat membuat fitur baru ini melalui **feature construction**? **Jawaban:**

1. **Feature Construction**: Luas apartemen bisa dihitung dengan mengalikan panjang dan lebar apartemen, yaitu:



1. **Alasan**: Dengan menambahkan fitur **Apartment Area** sebagai representasi dari ukuran fisik apartemen, model akan lebih mampu mengidentifikasi hubungan antara ukuran apartemen dengan harga sewa bulanan.

**Soal 3: Encoding pada Fitur Kategorikal**

**Soal:** Untuk mengoptimalkan input bagi model, fitur **Apartment Location** dan **Apartment Type** perlu dikonversi menjadi numerik. Apa teknik **encoding** yang sebaiknya digunakan untuk masing-masing fitur ini? **Jawaban:**

1. **Apartment Location**: Sebaiknya menggunakan **One-Hot Encoding** karena lokasi tidak memiliki urutan tertentu (nominal) dan setiap kategori harus diperlakukan secara independen oleh model.
2. **Apartment Type**: Jika jenis apartemen (Studio, 1-Bedroom, 2-Bedroom, 3-Bedroom) memiliki urutan tertentu atau nilai kategori yang dapat diberi makna urutan (ordinal), **Ordinal Encoding** bisa digunakan. Jika tidak, **One-Hot Encoding** lebih sesuai.

**Soal 4: Mengelompokkan Fitur Numerik sebagai Kategori**

**Soal:** Sebagai eksperimen tambahan, perusahaan ingin mengelompokkan nilai dari **Nearby Facilities Count** ke dalam tiga kategori: "Few", "Moderate", dan "Many". Bagaimana cara Anda melakukan **transformasi** ini, dan mengapa hal ini bermanfaat? **Jawaban:**

1. **Langkah Transformasi**: Kita dapat mengelompokkan nilai **Nearby Facilities Count** dengan rentang tertentu, misalnya:
   * "Few" untuk nilai 0-2
   * "Moderate" untuk nilai 3-5
   * "Many" untuk nilai di atas 5
2. **Alasan**: Dengan mengelompokkan fitur numerik ini menjadi kategori, model dapat mengenali pola hubungan yang lebih umum antara jumlah fasilitas di sekitar dan harga sewa, daripada terfokus pada nilai numerik yang spesifik. Ini membantu mencegah model menjadi terlalu kompleks dan membantu memperbaiki generalisasi.

**Soal 5: Feature Extraction dengan Teknik Vektorisasi pada Data Teks**

**Soal:** Misalkan dataset juga memiliki kolom **Apartment Description** yang berisi deskripsi teks mengenai apartemen. Jelaskan tiga langkah utama dalam **text feature extraction** untuk mengubah kolom ini menjadi fitur numerik. **Jawaban:**

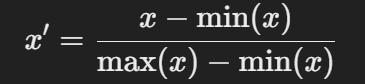
1. **Tokenisasi**: Pisahkan teks pada setiap kata, dengan spasi kosong dan tanda baca sebagai pembatas (misalnya, "apartemen luas dengan pemandangan kota" menjadi ["apartemen", "luas", "dengan", "pemandangan", "kota"]).
2. **Hitung Frekuensi**: Hitung jumlah kemunculan setiap kata (token) dalam deskripsi. Ini dapat dilakukan untuk setiap deskripsi apartemen.
3. **Normalisasi**: Mengurangi bobot kata-kata yang sering muncul di sebagian besar deskripsi (misalnya kata "apartemen" mungkin muncul di hampir semua deskripsi). Matriks frekuensi ini kemudian diubah menjadi **document-term matrix** atau **term-document matrix**.

Alasan: Mengubah teks menjadi data numerik membantu model memahami informasi tekstual tanpa memerlukan pemrosesan bahasa alami yang kompleks.

**Soal 6: Penerapan Data Transformation pada Fitur Numerik dengan Normalisasi**

**Soal:** Mengapa normalisasi pada fitur numerik, seperti **Apartment Length** dan **Apartment Width**, diperlukan sebelum memulai pelatihan model? Jelaskan salah satu metode normalisasi yang dapat digunakan. **Jawaban:**

* **Alasan Normalisasi**: Normalisasi membantu memastikan bahwa skala fitur numerik, seperti panjang dan lebar apartemen, tidak mendominasi dalam perhitungan algoritma, terutama ketika algoritma berbasis jarak (misalnya, KNN) atau berbasis gradien (misalnya, regresi linier).
* **Metode Normalisasi**: Salah satu metode yang umum digunakan adalah **Min-Max Scaling**, di mana nilai fitur dinormalisasi ke dalam rentang [0, 1]. Rumusnya adalah:



Dengan metode ini, nilai terendah pada fitur menjadi 0 dan nilai tertinggi menjadi 1.

**Soal 7: Mengapa Feature Construction Penting dalam Machine Learning?**

**Soal:** Jelaskan manfaat utama dari **feature construction** dalam pengembangan model machine learning, terutama dalam kasus dataset apartemen ini. **Jawaban:** **Manfaat Feature Construction**:

* **Menambah Informasi**: Dengan membuat fitur baru yang relevan, seperti **Apartment Area** dari panjang dan lebar, kita menyediakan informasi tambahan yang lebih bermakna bagi model untuk memahami hubungan yang lebih kompleks dalam data.
* **Meningkatkan Kinerja Model**: Fitur baru seringkali dapat menangkap pola yang lebih jelas, sehingga memudahkan model untuk membuat prediksi yang lebih akurat. Dalam kasus dataset apartemen ini, **Apartment Area** memberikan informasi yang lebih langsung mengenai ukuran keseluruhan, yang sering kali berkaitan dengan harga sewa.

**Soal 8: Membuat Fitur Kategori dari Fitur Numerik (Binning)**

**Soal:** Misalkan Anda ingin mengubah fitur **Monthly Rent Price** menjadi kategori berdasarkan tingkat harga ("Low", "Medium", "High"). Jelaskan bagaimana Anda akan melakukan proses **binning** ini dan sebutkan manfaatnya. **Jawaban:**

* **Proses Binning**: Kita dapat membuat tiga kategori berdasarkan rentang harga. Misalnya:
  + **Low**: di bawah Rp 5 juta
  + **Medium**: Rp 5 juta hingga Rp 10 juta
  + **High**: di atas Rp 10 juta
* **Manfaat**: Dengan mengubah fitur numerik menjadi kategori, kita mengurangi kompleksitas dari fitur tersebut, yang bisa berguna untuk model yang memerlukan klasifikasi umum daripada nilai presisi. Ini juga memudahkan dalam segmentasi data dan mengidentifikasi pola umum berdasarkan rentang harga.

**Soal 9: Identifikasi Teknik Feature Extraction yang Tepat untuk Deskripsi Teks Apartemen**

**Soal:** Dalam dataset, kolom **Apartment Description** berisi teks yang menggambarkan fitur apartemen. Pilih salah satu teknik **feature extraction** untuk data teks ini dan jelaskan mengapa teknik tersebut sesuai. **Jawaban:**

* **Teknik yang Dipilih**: **Bag of Words (BoW)**, di mana setiap kata yang muncul dalam deskripsi diubah menjadi kolom dalam matrix dan dihitung frekuensinya dalam setiap dokumen.
* **Alasan Pemilihan Teknik**: BoW adalah teknik sederhana dan efektif untuk kasus di mana kita hanya memerlukan informasi frekuensi kemunculan kata-kata untuk mengenali pola. Teknik ini membantu model mengenali kata-kata yang sering muncul dalam deskripsi (seperti "nyaman" atau "strategis") dan dapat membantu mengidentifikasi fitur yang berhubungan dengan preferensi penyewa.

**Soal 10: Mengelola Nilai Hilang dalam Dataset**

**Soal:** Anda menemukan bahwa beberapa apartemen dalam dataset tidak memiliki informasi lengkap mengenai **Nearby Facilities Count**. Bagaimana Anda dapat menangani nilai hilang ini untuk memastikan kualitas data tetap terjaga? Sebutkan dua metode penanganan nilai hilang dan keunggulannya.

**Jawaban:**

1. **Mengisi dengan Mean atau Median**:
   * **Cara Kerja**: Isi nilai hilang dengan rata-rata atau nilai tengah dari kolom tersebut.
   * **Keunggulan**: Metode ini menjaga distribusi data dan mengurangi risiko bias. Ini sangat efektif jika fitur memiliki distribusi yang normal atau mendekati normal.
2. **Mengisi dengan Kategori Spesifik** (seperti "Tidak Diketahui" atau "0" jika sesuai):
   * **Cara Kerja**: Buat kategori baru yang mewakili "Tidak Diketahui" untuk data yang hilang atau isikan dengan nilai 0 jika absennya fasilitas dianggap berarti.
   * **Keunggulan**: Pendekatan ini menjaga data tetap utuh tanpa asumsi statistik, dan berguna ketika nilai yang hilang tidak terlalu banyak atau memiliki makna khusus.

Berikut adalah materi yang mendalam dan terstruktur tentang konsep-konsep utama untuk UTS Anda dalam feature engineering:

**1. Feature Label dan Variable Label**

**Feature Label**:

* Feature adalah atribut atau kolom pada dataset yang digunakan untuk memberi informasi yang relevan ke dalam model machine learning. Contoh fitur meliputi **panjang apartemen**, **harga sewa**, **jumlah kamar tidur**, dll.
* **Feature Label** mengacu pada nilai atau kategori yang dimiliki oleh sebuah fitur dalam dataset. Misalnya, dalam fitur **Tipe Apartemen**, label dapat berupa "Studio", "1 Kamar Tidur", atau "2 Kamar Tidur".
* Feature-label merupakan bagian dari representasi data yang menjadi input dalam proses pembelajaran mesin.

**Variable Label**:

* **Variable Label** merujuk pada target atau keluaran yang ingin diprediksi oleh model machine learning. Ini sering kali merupakan kolom dalam dataset yang merepresentasikan tujuan prediksi, seperti **harga apartemen** atau **penjualan produk**.
* Dalam supervised learning, variable label merupakan variabel dependen yang dipengaruhi oleh variabel bebas (fitur-fitur) dalam model. Misalnya, ketika memprediksi harga apartemen, harga tersebut adalah variable label yang dipengaruhi oleh berbagai fitur lain (seperti luas dan lokasi).
* Identifikasi feature label dan variable label sangat penting dalam menentukan input dan output model serta mempersiapkan data yang benar untuk pelatihan.

**2. Feature Transformation**

Feature transformation adalah proses mengubah atau memodifikasi fitur untuk membuat data lebih optimal dan efektif digunakan dalam model pembelajaran mesin. Tujuannya adalah:

1. Meningkatkan representasi fitur untuk meningkatkan kinerja model.
2. Mengurangi noise atau karakteristik yang tidak relevan dalam data.
3. Mengubah format data menjadi lebih mudah diterima oleh algoritma pembelajaran mesin.

Feature transformation memiliki dua metode utama:

**a. Feature Construction**

* Feature construction adalah proses menciptakan fitur baru dari fitur yang sudah ada untuk mendapatkan lebih banyak informasi atau representasi yang lebih baik dari data.
* **Contoh**: Dari fitur **panjang** dan **lebar apartemen**, kita bisa membangun fitur baru **luas apartemen** dengan cara menghitung hasil perkalian panjang dan lebar.
* Fitur yang baru ini sering kali memberikan wawasan tambahan dan dapat membuat model lebih efektif karena fitur baru tersebut lebih langsung menggambarkan karakteristik yang relevan.

**Keunggulan Feature Construction**:

* **Meningkatkan Akurasi Model**: Fitur baru yang dibuat bisa lebih sesuai dengan masalah atau pertanyaan yang ingin dijawab.
* **Mengurangi Dimensi Data**: Dengan memilih dan menciptakan fitur yang lebih bermakna, kita dapat menyederhanakan data sehingga lebih mudah diinterpretasikan.

**b. Feature Extraction**

* Feature extraction adalah proses memadatkan informasi dari fitur-fitur asli menjadi fitur yang lebih sedikit tetapi tetap representatif.
* **Contoh**: Dalam analisis teks, kita bisa menggunakan metode **Bag of Words (BoW)** atau **TF-IDF** untuk mengekstraksi fitur dari kata-kata dalam teks. Ini menghasilkan vektor angka yang merepresentasikan frekuensi atau bobot kemunculan kata dalam dokumen.
* Dalam feature extraction, fitur yang sangat banyak dipadatkan menjadi sejumlah fitur yang memiliki bobot atau representasi yang lebih terfokus.

**Keunggulan Feature Extraction**:

* **Mengurangi Dimensi Data**: Data menjadi lebih padat dan terstruktur, sehingga lebih cepat diproses oleh model.
* **Mencegah Overfitting**: Dengan membatasi jumlah fitur, model lebih fokus pada informasi yang penting saja dan tidak “terjebak” dalam detail yang tidak perlu.

**3. Data Transformation (Data Extraction & Data Construction)**

Data transformation mencakup segala proses modifikasi fitur agar lebih sesuai dengan kebutuhan machine learning, baik dari sisi analisis maupun interpretasi data. Dalam feature engineering, transformasi data meliputi beberapa metode penting seperti normalisasi, encoding, binning, dan sebagainya.

**Data Extraction**:

* Proses ini menitikberatkan pada teknik untuk menarik atau mengisolasi data-data penting yang dapat digunakan dalam analisis.
* **Contoh**: Menarik teks atau gambar dari internet untuk kemudian diolah menjadi dataset, atau mengekstraksi kata-kata kunci dari dokumen teks.

**Data Construction**:

* Data construction mengacu pada proses penambahan atau pengisian informasi dalam data yang sudah ada untuk meningkatkan keakuratan atau relevansi data.
* **Contoh**: Ketika menggunakan dataset customer feedback, kita dapat membuat label sentimen sebagai konstruksi tambahan yang membantu model memahami konteks positif atau negatif dari teks.

**Contoh Praktis Proses dalam Feature Engineering**

Untuk membantu memahami penerapan teknik di atas, berikut ini adalah beberapa contoh proses:

1. **Identifikasi Feature Label dan Variable Label**:
   * **Dataset**: Misalkan, dalam dataset properti, kita memiliki fitur **panjang apartemen** dan **jumlah kamar** sebagai feature label, sementara **harga sewa** sebagai variable label (target yang ingin diprediksi).
2. **Feature Construction**:
   * **Langkah**: Dari fitur **panjang** dan **lebar apartemen**, kita dapat membuat fitur **luas apartemen** dengan rumus matematika (panjang x lebar).
   * **Manfaat**: Fitur luas lebih menggambarkan ukuran properti secara langsung, yang lebih relevan untuk prediksi harga sewa.
3. **Feature Extraction**:
   * **Langkah**: Menggunakan vektorisasi (BoW) untuk data teks dalam fitur **deskripsi apartemen**. Setiap kata diubah menjadi vektor angka yang bisa dihitung kemunculannya.
   * **Manfaat**: Representasi angka memudahkan data teks diolah dalam model machine learning.
4. **Normalisasi (Data Transformation)**:
   * **Langkah**: Fitur seperti **Apartment Length** atau **Apartment Width** bisa dinormalisasi ke skala [0, 1] menggunakan metode **Min-Max Scaling**.
   * **Manfaat**: Normalisasi membuat skala fitur konsisten dan menghindari dominasi satu fitur tertentu, membantu model lebih stabil dalam pelatihan.

**Kesimpulan**

Feature engineering, melalui proses label assignment, feature transformation (feature construction & feature extraction), serta data transformation, berperan krusial dalam persiapan data untuk machine learning. Teknik-teknik ini bertujuan untuk mengidentifikasi fitur yang paling relevan, membuat fitur baru yang lebih informatif, dan mengubah format data agar lebih sesuai dengan algoritma yang digunakan. Dengan pemahaman mendalam tentang setiap teknik dan metode ini, kita dapat menciptakan dataset yang lebih kuat dan relevan untuk pelatihan model.

**Normalisasi Fitur Numerik**

**Normalisasi fitur numerik** adalah teknik dalam feature engineering untuk mengubah skala data numerik menjadi rentang nilai tertentu, umumnya antara 0 dan 1. Proses ini penting karena model machine learning biasanya bekerja lebih baik jika data berada dalam skala yang konsisten. Tanpa normalisasi, fitur dengan rentang nilai yang lebih besar bisa mendominasi model, menyebabkan hasil yang kurang optimal.

**Pengertian Normalisasi**

Normalisasi adalah proses mengubah nilai fitur numerik agar berada dalam skala yang seragam, misalnya dengan membatasi nilai fitur pada interval tertentu, seperti [0, 1]. Dengan kata lain, normalisasi menghindari adanya fitur yang memiliki skala jauh lebih besar daripada fitur lainnya, yang dapat memengaruhi proses pelatihan model.

**Langkah-langkah Normalisasi**

1. **Pahami Rentang Data**: Sebelum melakukan normalisasi, penting untuk mengetahui rentang data asli pada fitur numerik. Misalnya, nilai fitur bisa bervariasi antara 0 dan 100, atau bisa juga antara -100 dan 1000.
2. **Pilih Teknik Normalisasi**: Salah satu teknik yang paling umum digunakan adalah Min-Max Normalization, yang merubah nilai fitur menjadi rentang [0, 1]. Formula untuk teknik ini adalah:

A black background with white text

Description automatically generated

Di mana:

* + XXX adalah nilai fitur numerik yang akan dinormalisasi,
  + min⁡(X)\min(X)min(X) adalah nilai minimum dari fitur tersebut,
  + max⁡(X)\max(X)max(X) adalah nilai maksimum dari fitur tersebut,
  + XnormX\_{\text{norm}}Xnorm​ adalah nilai fitur yang sudah dinormalisasi.

1. **Terapkan Normalisasi**: Setelah menentukan rentang data dan memilih teknik yang sesuai, kita bisa menerapkan normalisasi menggunakan metode yang sesuai. Misalnya, di Python menggunakan library scikit-learn ada MinMaxScaler() yang melakukan normalisasi fitur ke dalam rentang [0, 1].

**Manfaat Normalisasi**

* **Menghindari Skala Dominasi**: Dengan normalisasi, fitur yang memiliki rentang nilai besar tidak mendominasi fitur lain yang memiliki rentang kecil, sehingga model bisa belajar lebih adil.
* **Mempercepat Konvergensi Model**: Beberapa algoritma machine learning, seperti Gradient Descent, lebih cepat dalam menemukan solusi optimal ketika data sudah dinormalisasi.

**Encoding Fitur Kategorikal Menggunakan One-Hot Encoding**

**One-hot encoding** adalah teknik untuk mengubah data kategorikal menjadi format numerik yang bisa dipahami oleh model machine learning. Dalam teknik ini, setiap kategori dari fitur kategorikal diubah menjadi sebuah vektor biner (0 atau 1). Setiap kategori dipetakan ke dalam kolom terpisah, dan hanya satu nilai yang diatur ke 1, sementara sisanya diatur ke 0.

**Pengertian One-Hot Encoding**

One-hot encoding adalah proses yang mengubah setiap nilai dalam fitur kategorikal menjadi representasi biner. Misalnya, jika kita memiliki kolom "warna" dengan kategori "merah", "hijau", dan "biru", one-hot encoding akan mengubahnya menjadi tiga kolom: satu untuk "merah", satu untuk "hijau", dan satu untuk "biru". Setiap baris kemudian akan diubah menjadi vektor yang hanya memiliki satu nilai 1 pada kolom yang sesuai dengan kategori dan sisanya 0.

**Langkah-langkah One-Hot Encoding**

1. **Identifikasi Fitur Kategorikal**: Tentukan kolom mana yang mengandung data kategorikal, misalnya "warna", "kota", atau "jenis kelamin".
2. **Buat Kolom untuk Setiap Kategori**: Misalnya, untuk fitur kategorikal "warna" yang memiliki tiga kategori yaitu "merah", "hijau", dan "biru", kita akan membuat tiga kolom baru:
   * Kolom "merah"
   * Kolom "hijau"
   * Kolom "biru"
3. **Masukkan Nilai 1 atau 0**: Untuk setiap entri dalam dataset, kolom yang sesuai dengan kategori yang ada akan diberi nilai 1, sementara kolom lain diberi nilai 0. Sebagai contoh:
   * Jika data menunjukkan "merah", maka kolom "merah" diisi 1, dan kolom "hijau" serta "biru" diisi 0.
   * Jika data menunjukkan "hijau", maka kolom "hijau" diisi 1, dan kolom lainnya diisi 0.

**Fungsi dari One-Hot Encoding**

* **Memungkinkan Model Memahami Data Kategorikal**: Machine learning model tidak dapat langsung bekerja dengan data kategorikal yang berisi teks. One-hot encoding mengubah teks menjadi format numerik yang bisa diproses oleh model.
* **Menghindari Urutan Tidak Relevan**: One-hot encoding menghindari kesalahan interpretasi dalam beberapa teknik encoding lain (misalnya label encoding), di mana model bisa saja menganggap ada urutan atau hubungan numerik antara kategori yang berbeda, padahal itu tidak berlaku dalam konteks data.

**Manfaat One-Hot Encoding**

1. **Meningkatkan Akurasi Model**: One-hot encoding memungkinkan model untuk belajar dari data kategorikal tanpa memaksa model untuk menganggap kategori tersebut memiliki hubungan ordinal (urutan). Ini membuat prediksi model menjadi lebih akurat.
2. **Kompatibilitas dengan Algoritma Machine Learning**: Banyak algoritma machine learning, terutama yang berbasis regresi atau pohon keputusan, membutuhkan fitur dalam format numerik. One-hot encoding memastikan bahwa data kategorikal bisa digunakan oleh berbagai algoritma.
3. **Mencegah Misinterpretasi Model**: Dengan one-hot encoding, model tidak akan menganggap kategori dengan nilai numerik lebih tinggi lebih penting atau lebih besar daripada kategori lainnya. Semua kategori diperlakukan sama.

**Kesimpulan**

**Normalisasi fitur numerik** dan **one-hot encoding** adalah dua teknik dasar namun sangat penting dalam *feature engineering*. Normalisasi membantu menyamakan skala data numerik agar model machine learning tidak terpengaruh oleh rentang yang berbeda antar fitur. Sementara itu, **one-hot encoding** memungkinkan data kategorikal diubah menjadi format yang bisa dipahami oleh model, dengan memastikan bahwa kategori-kategori tersebut tidak dianggap memiliki hubungan ordinal.

Dengan memahami kedua teknik ini dan menerapkannya dengan benar, kita dapat meningkatkan kualitas model machine learning yang dibangun dan memastikan bahwa model tersebut dapat berfungsi dengan baik dalam prediksi yang lebih akurat.