Pendeteksian Citra Pengunjung Menggunakan *Single Shot Detector* untuk Analisis dan Prediksi *Seasonality*

http://dx.doi.org/10.28932/jutisi.v7i1.3329
Riwayat Artikel
Received: 24 Januari 2021 | *Final Revision*: 4 Maret 2021 | *Accepted*: 12 Maret 2021

Deon Diamanta[#1], Hapnes Toba[✉*2]

[#]*Program Studi Magister Ilmu Komputer, Fakultas Teknologi Informasi*
Universitas Kristen Maranatha
Jl. Surya Sumantri no. 65, Bandung
[1]mi1979004@student.it.maranatha.edu
[2]hapnestoba@it.maranatha.edu

Abstract — **This study discusses the analysis of retail store with time series method to obtain information about sales trend and seasonality by looking at visitor data and total transaction data at a time period. Data in the form of the number of customers who visit are obtained through CCTV video camera recordings placed at retail store X and the total transaction occurred at retail store X. The visitor counting uses the deep learning method with SSD (Single Shot Detector) object detection framework and MobileNet architecture. The library used to count the number of customers visiting the store is OpenCV, Pandas, Numpy, Dlib, and Imutils. The number of customers visiting the store will then be compared to the number of transactions that occur at the same time so that a conversion rate can be obtained. From here, we can see sales trend that occur at any time. Time series analysis is also carried out to determine and analyze the pattern of data obtained based on certain time to predict the things that need to be done in the future. Through this research, information has been successfully obtained related to seasonality patterns, value and interpretation of retail conversion rates, models for predicting the number of visitors and transactions, and answering the hypothesis with the Wilcoxon test method obtained a p-value of 0,014 which states that the data pattern of the number of consumers is not the same as the transaction data pattern.**

Keywords— **computer vision; MobileNet, OpenCV, people counting, Single Shot Detector, time series analysis.**

I. PENDAHULUAN

Apa masa depan dari *retail*? Meskipun kita masih belum mengetahuinya dan tidak ada bola kristal ajaib yang dapat memprediksi apa yang akan terjadi di masa yang akan datang, dapat dilakukan penelitian dan ada tren yang dapat membantu pelaku usaha *retail* untuk mendapatkan berbagai keunggulan dalam kompetisi. Dalam usaha mencapai pengembangan bisnis, banyak cara yang dilakukan oleh pelaku usaha *retail* seperti system penghitungan stok barang, menggunakan beragam media pemasaran untuk mengundang calon pelanggan datang berbelanja ke toko, penerapan *customer's feedback*, hingga penggunaan *clicker manual* sederhana untuk menghitung jumlah pengunjung yang datang mengunjungi toko *retail* (*footfall*) [1].

Semua hal tersebut, terutama penghitungan pengunjung dilakukan oleh pelaku usaha *retail* karena memiliki tujuan penting, yaitu memberi tahu pelaku usaha *retail* seberapa sukses strategi pemasaran yang dilakukan, dalam hal menarik perhatian calon pelanggan. Banyaknya orang yang masuk ke dalam toko merupakan indikator bahwa iklan, kampanye pemasaran, maupun etalase / penempatan produk dalam toko secara efektif mampu menarik banyak calon pelanggan untuk datang mengunjungi toko *retail*. Sebaliknya, jumlah pengunjung yang rendah dapat menunjukkan kampanye pemasaran yang kurang berhasil [2].

Selain penghitungan pengunjung, pelaku usaha *retail* juga perlu memperhatikan tingkat konversi. Dalam konteks *retail*, konversi adalah proses mengubah calon pelanggan (misalnya orang yang datang mengunjungi toko *retail*) menjadi pembeli yang melakukan transaksi. Untuk mengetahui tingkat konversi, seorang pelaku usaha *retail* harus terlebih dahulu mengetahui berapa banyak pengunjung toko setiap harinya. Mengetahui jumlah transaksi yang terjadi pada setiap hari saja tidak cukup. Tanpa mengetahui seberapa banyak orang yang datang mengunjungi toko, maka pelaku usaha *retail* kehilangan bagian penting dari data untuk mengukur performa toko [3].

Dengan adanya dukungan teknologi yang berkembang pesat

seperti pada saat ini, pelaku usaha *retail* dapat memanfaatkan teknologi seperti *computer vision* untuk melakukan tugas penghitungan pengunjung yang pada awalnya menggunakan alat *clicker manual*. Kegiatan menghitung pengunjung ini merupakan sebuah aktivitas sederhana, namun memerlukan banyak waktu. *Computer vision* merupakan suatu bidang ilmiah interdisipliner yang membahas dan mempelajari bagaimana komputer dapat memperoleh pemahaman tingkat tinggi dari sumber berupa gambar atau *video digital* [4].

Dari perspektif teknis, *computer vision* berusaha memahami dan mengotomatisasi tugas-tugas yang dapat dilakukan oleh sistem *visual* manusia, Sistem penghitungan pengunjung di lingkungan *retail* digunakan untuk menghitung tingkat konversi, yang merupakan persentase dari total pengunjung dibandingkan dengan jumlah transaksi yang terjadi dalam suatu kurun waktu tertentu. Tingkat konversi seringkali menunjukkan potensi dan juga performa secara umum dari sebuah toko *retail* [5].

Latar belakang penelitian ini, dalam mengembangkan perusahaan, pelaku usaha *retail* tidak bisa hanya mengandalkan intuisi semata. Perlu adanya dukungan data yang dapat memberikan pandangan baru untuk memastikan langkah yang akan dipilih merupakan hal yang tepat untuk dilakukan. Sekarang ini, teknologi banyak digunakan oleh berbagai bidang bisnis untuk meningkatkan efisiensi dan juga ketepatan dalam menjalankan proses operasional sehari-hari. Banyak perkembangan teknologi yang sebelumnya belum pernah digunakan dalam bisnis *retail* seperti pada studi ini, terutama di Indonesia.

Salah satu contoh teknologi yang dimaksud adalah *computer vision*. Teknologi seperti ini jika dimanfaatkan pada bidang bisnis *retail* akan memberikan sudut pandang baru dalam menentukan keputusan pengembangan bisnis berikutnya. Teknologi-teknologi seperti itu dapat membantu pelaku usaha untuk mengukur performa bisnisnya. Oleh karena itu, penelitian ini dilakukan untuk mencari tahu mengenai pemanfaatan teknologi semacam itu dalam membantu performa toko *retail*.

Mendeteksi, melacak, dan menghitung orang yang masuk atau melewati ruangan menjadi alat penting untuk penelitian statistik perusahaan yang bergerak pada bidang *retail*. Dalam penelitian ini, rumusan masalah yang ingin dicari pemecahannya adalah bagaimana memanfaatkan pendeteksian citra pada pintu masuk toko *retail* untuk mengetahui pola tren, *seasonality*, dan tingkat konversi toko.

Tujuan penelitian ini adalah mengidentifikasi pola tren dan *seasonality* melalui jumlah penghitungan pengunjung dengan memanfaatkan pendeteksian citra (*computer vision*) dan jumlah transaksi dalam suatu kurun waktu tertentu pada toko *retail X*, kemudian jumlah penghitungan pengunjung akan dibandingkan dengan data jumlah transaksi dalam kurun waktu yang sama untuk memperoleh tingkat konversi yang menjadi informasi terkait performa toko tersebut.

Penelitian ini dilakukan untuk mencari tahu mengenai *seasonality* pengunjung yang diperoleh dari data video *cctv* toko *retail X* dalam kurun waktu sekitar 3 minggu. Penelitian ini diharapkan mampu menjawab hipotesis yang ingin dicari tahu kebenarannya, yaitu:

- H0: Data jumlah pengunjung toko *retail X* menunjukkan pola *seasonality* yang serupa dengan pola data transaksi;
- H1: Data jumlah pengunjung toko *retail X* menunjukkan pola *seasonality* yang berbeda dengan pola data transaksi.

Ruang lingkup yang dimiliki pada penelitian yaitu dalam penelitian ini dibahas mengenai pemanfaatan teknologi *computer vision* untuk memperoleh jumlah pengunjung toko *retail* dari data *video cctv* yang terdapat pada toko *retail X*. Data jumlah pengunjung kemudian akan dibandingkan dengan data jumlah transaksi dalam kurun waktu yang sama untuk memperoleh tingkat konversi toko *retail X*. Data-data yang diperoleh tersebut kemudian akan diproses dengan metode *time series* untuk mendapatkan gambaran mengenai pola tren dan *seasonality* yang terjadi. Toko *retail X* memiliki jalur akses keluar masuk yang memungkinkan untuk adanya penempatan kamera *cctv*. Data yang digunakan hanya terkait data transaksi saja dan tidak mencakup data lainnya.

II. LANDASAN TEORI

Dalam penelitian ini, menggunakan teori-teori yang terkait dengan *computer vision* untuk melakukan pendeteksian orang dan *time series analysis* untuk memperoleh gambaran pola tren dan *seasonality*.

A. Computer Vision

Computer vision, atau seringkali disingkat sebagai CV, adalah suatu bidang studi yang mengembangkan teknik untuk membantu komputer "melihat" serta memahami konsep gambar *digital* seperti foto dan *video*. *Computer vision* berupaya untuk mereplikasi kemampuan penglihatan biologis manusia bagi komputer. Terlihat sederhana, hanya saja terdapat kompleksitas dalam melakukan hal tersebut sehingga seringkali menjadi hal yang masih belum terpecahkan. Namun, dengan perkembangan dalam kecerdasan buatan, *computer vision* mampu berkembang dalam beberapa tahun terakhir dan telah mampu melampaui manusia dalam beberapa tugas yang berkaitan dengan mendeteksi objek [6].

Computer vision adalah ekstraksi informasi secara otomatis dari gambar. Informasi dapat berarti apa saja mulai dari data gambar dari model 3D, data gambar yang diperoleh dari berbagai sudut pandang kamera, deteksi dan pengenalan objek, hingga pengelompokkan dan pencarian konten gambar [7]. *Computer vision* akan mengolah data berupa gambar dan *video* yang disebut dengan *image processing*. *Image processing* adalah proses pembuatan gambar baru dari gambar yang ada. Biasanya dengan menyederhanakan atau meningkatkan konten dengan cara-cara tertentu. Hal ini merupakan jenis pemrosesan sinyal *digital* dan tidak peduli dengan memhami konten suatu gambar, *image processing* meliputi:

- Menormalkan sifat fotometrik gambar, seperti kecerahan atau warna;
- Memotong batas gambar, seperti memusatkan objek pada foto;
- Menghapus *noise digital* dari suatu gambar, seperti artefak *digital* dan tingkat cahaya rendah.

Dengan adanya kemajuan pada bidang *computer vision*, dapat digunakan untuk membangun sistem yang mampu mengekstraksi informasi yang berguna dari gambar. Hingga kini, pemanfaatan *computer vision* dapat memberikan keuntungan dan membantu berbagai permasalahan tingkat tinggi dari berbagai bidang seperti [8]:

- *Optical Reader Recognition* (OCR);
- *Machine Inspection;*
- *Retail* (*Automated Checkouts*);
- *3D Model Building;*
- *Medical Imaging;*
- *Automotive safety;*
- *Motion capture;*
- *Surveillance;*
- *Fingerprint recognition and biometrics.*

Computer vision banyak diaplikasikan dalam tugas-tugas yang berkaitan dengan mengenali hal-hal dalam gambar, sebagai contoh:

- *Object classification*: apa kategori objek yang terdapat dalam gambar?
- *Objek identification*: jenis objek apa yang terdapat pada gambar?
- *Object verification*: apa objek yang dimaksud berada dalam gambar?
- *Object detection*: di mana posisi objek berada pada gambar?
- *Object segmentation*: piksel apa yang dimiliki oleh objek pada gambar?
- *Object recognition*: objek apa yang berada pada gambar dan di mana posisinya?

B. OpenCV

OpenCV (*Open Source Computer Vision Library*) adalah perpustakaan perangkat lunak yang bersifat *open source* untuk pembelajaran mesin dan *computer vision*. *OpenCV* dibangun untuk menyediakan infrastruktur umum untuk aplikasi *computer vision* dan untuk membantu persepsi mesin dalam memproses gambar serta *video* untuk mengidentifikasi wajah, objek, dan tulisan tangan manusia [9]. Tidak hanya terbatas pada ketiga hal tersebut, *OpenCV* juga dapat digunakan untuk mengklasifikasikan tindakan manusia, melacak gerakan benda, mengekstraksi model objek 3D, menggabungkan beberapa gambar menjadi satu kesatuan untuk menghasilkan gambar seluruh adegan beresolusi tinggi, menemukan gambar serupa dari basis data gambar, hingga membuat lapisan *augmented reality*.

OpenCV memiliki antarmuka C++, *Python*, *Java*, dan *MATLAB* serta mendukung sistem operasi *Windows, Linux, Android, dan Mac OS*. *OpenCV* sebagian besar condong ke aplikasi penglihatan *realtime* dan memanfaatkan instruksi MMX dan SSE saat tersedia. CUDA dan antarmuka OpenCL berfitur lengkap sedang dikembangkan secara aktif saat ini [10]. Selain pemanfaatan *OpenCV* yang telah disebutkan di atas, *OpenCV* juga dapat digunakan untuk melacak dan menghitung objek berupa orang yang datang berkunjung ke toko.

C. Deteksi Objek

Penghitungan orang untuk memperkirakan tingkat kesuksesan suatu acara atau tingkat performa suatu toko *retail* sudah menjadi hal yang perlu dilakukan oleh setiap pelaku usaha *retail* saat ini. Hal ini dapat dilakukan dengan menggunakan sistem penghitungan pengunjung dengan data *video cctv* yang ditempatkan pada area pintu masuk toko *retail*, dengan tujuan memperoleh data statistik yang memberikan *insight* mengenai keberhasilan dari tempat tersebut [11]. Dalam memperoleh data statistik tersebut, bukanlah suatu hal yang mudah, karena memerlukan posisi penempatan kamera *cctv* yang disesuaikan dengan kebutuhan penghitungan orang. Posisi kamera harus berada di area atas pintu masuk dan menghadap kebawah (*overhead position*) untuk merekap pergerakan orang yang memasuki toko [12].

Tujuan penempatan dan pengaturan posisi kamera yang seperti itu adalah untuk mempermudah proses penghitungan orang-orang yang melewati area pandang kamera secara bersamaan. Selain penempatan dan posisi kamera, perlu juga menambahkan garis virtual dengan kategori masuk dan keluar (*up & down*) untuk mengetahui dengan jelas mana orang yang masuk ke dalam toko dan mana orang yang berjalan keluar dari toko [13]. Ada hal lainnya yang juga penting dalam penghitungan orang, yaitu deteksi objek untuk mendeteksi benda berupa orang yang datang ke toko *retail* dan direkam melalui kamera *cctv*.

Melalui pemanfaatan deteksi objek, dapat memberikan koordinat (x, y) *bounding box* pada objek yang terdeteksi dalam gambar. Deteksi objek tidak hanya memberitahukan benda apa yang ada pada gambar, tetapi juga memberitahukan di mana posisi benda tersebut dalam gambar. Dalam *deep learning*, terdapat tiga jenis *framework* deteksi objek yang populer dan sering digunakan:

- *Faster R-CNNs*
- *You Only Look Once (YOLO)*
- *Single Shot Detectors (SSDs)*

Faster R-CNNs kemungkinan adalah *framework* deteksi objek yang paling sering didengar untuk *deep learning*. Namun, tekniknya sangat sulit untuk dipahami (terutama untuk pemula) dan juga cukup sulit untuk diimplementasikan. Terlebih metode ini bisa sangat berat secara komputasi, di mana performa yang diperoleh hanya berkisar di 7 FPS (*frame per second*). Lain halnya dengan *framework YOLO* yang lebih cepat jika dibandingkan dengan *R-CNNs*, bisa menghasilkan performa deteksi objek hingga kisaran 40-90 FPS dengan menggunakan

perangkat komputasi yang memadai seperti Titan X GPU. Hanya saja *YOLO* memiliki akurasi yang kurang dari *R-CNN*. Kemudian yang terakhir ada *SSDs*, awal mulanya dikembangkan oleh *Google*. *SSD* merupakan *framework* deteksi objek yang berada di antara *R-CNN* dan *YOLO* (*balance*), memiliki karakteristik performa FPS yang cukup cepat dan cenderung lebih akurat jika dibandingkan dengan *framework YOLO* [14].

D. Retail Conversion Rate

Retail conversion rate adalah proporsi dari jumlah pengunjung ke toko *retail* dengan jumlah transaksi yang dilakukan pada suatu kurun waktu yang sama. Dengan *OpenCV*, dapat melakukan *people counter* untuk toko *retail*. *People counter* ini bertujuan untuk memperoleh perhitungan jumlah orang yang berkunjung yang kemudian dibandingkan dengan proporsi orang yang melakukan transaksi. Jika ada 300 orang datang mengunjungi toko dalam sehari, tetapi hanya 75 yang melakukan transaksi, maka tingkat konversi adalah sebesar 25 persen [15]. *Retail conversion rate* memberitahu pelaku usaha bisnis *retail* seberapa baik yang telah mereka lakukan dalam mengubah pengunjung menjadi pembeli yang melakukan transaksi. Nilai *retail conversion rate* memberikan wawasan kritis tentang apa yang terjadi di dalam toko.

Tanpa mengukur data data seperti ini, keputusan bisnis pada dasarnya hanya didasarkan pada dugaan atau insting semata. Seringkali orang yang berkunjung ke toko, pada akhirnya tidak melakukan transaksi. Hal tersebut merupakan suatu hal umum yang bisa terjadi pada siapa saja. Seluruh toko *retail* manapun, tidak mungkin ada yang bisa membuat seluruh pengunjung menjadi pembeli. Besarnya *retail conversion rate* juga akan berbeda-beda bagi setiap jenis bisnis / toko *retail* [16]. Mengukur *retail conversion rate* memberikan gambaran tentang seberapa baik performa toko, dibandingkan hanya melihat angka penjualan semata. Menghitung *retail conversion rate* akan memberi tahu para pelaku usaha bisnis *retail* seberapa banyak prospek pengunjung yang berubah menjadi penjualan aktual (transaksi) – yang dapat diartikan sebagai lebih banyak uang yang dihasilkan untuk bisnis [17].

E. Time Series Analysis

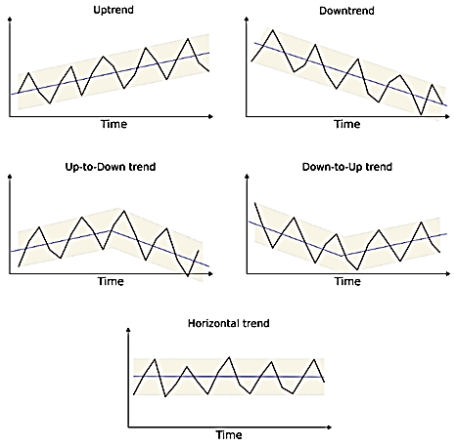
Time series analysis adalah penggunaan metode statistik untuk melakukan prediksi perilaku / kondisi masa depan berdasarkan data historis. Ini mirip dengan pendekatan pembelajaran statistik lainnya, seperti *supervised* dan *unsupervised learning*. Namun, *time series analysis* memiliki banyak hal yang membuatnya berbeda dari *machine learning* biasa, seperti pemrosesan data hingga memodelkan validasi [18]. Banyak perusahaan yang mengeksplorasi *time series analysis* sebagai cara untuk membuat keputusan bisnis yang lebih baik. Sebagai contoh bagi toko *retail*, jika memiliki *insight* mengenai berapa banyak konsumen yang akan datang pada pertengahan tahun, toko *retail* dapat merencanakan jumlah *staff*, jenis promosi, dan *event* yang pas.

Time series merupakan urutan titik data yang direkam berdasarkan waktu. Jadi, ketika berhadapan dengan data *time series*, urutan menjadi hal penting. Secara khusus, nilai-nilai dalam *time series* mengungkapkan ketergantungan terhadap waktu. Akibatnya, jika urutan *time series* berubah, maka makna datanya pun ikut berubah. Biasanya, data *time series* memiliki dua sifat penting:

- Data diukur secara berurutan dan memiliki jarak waktu yang sama;
- Setiap unit waktu memiliki paling banyak satu pengukuran data

Dalam *time series analysis*, waktu seringkali merupakan variabel independen dan tujuan dari analisis data terkait biasanya untuk mengidentifikasi pola yang menjelaskan perilaku data *time series* dan membuat ramalan masa depan berdasarkan pola-pola tersebut [19]. Sebagian besar data *time series* biasanya memiliki setidaknya satu dari tiga jenis pola ini: tren, *seasonality*, dan *cycle*. Tren menggambarkan perilaku umum suatu data *time series*. Jika data rangkaian waktu memanifestasikan kemiringan jangka panjang yang positif dari waktu ke waktu, maka data rangkaian waktu tersebut memiliki tren naik.

Jika sebaliknya, data rangkaian waktu menggambarkan kemiringan negatif secara umum, maka data rangkaian waktu tersebut memiliki tren menurun. Tren keseluruhan juga dapat berubah arah. Bisa ada tren naik turun atau tren turun naik. Tren stasioner atau *horizontal* mendefinisikan data *time series* dengan pola jangka panjang positif maupun negatif seperti pada gambar 1 berikut.



Gambar 1. Tren dalam data *time series*

Pola *seasonal* adalah segala jenis fluktuasi (perubahan) dalam *time series* yang disebabkan oleh peristiwa yang memiliki sifat musiman. Peristiwa ini dapat berupa waktu yang terjadi berulang pada setiap tahun (musim hujan atau musim panas), atau waktu harian (hari Sabtu yang selalu ada setiap minggu). *Seasonal* selalu memiliki frekuensi tetap, pola *seasonal* selalu dimulai dan berakhir pada periode yang sama. Sebagai contoh, toko pakaian yang menjual mantel tebal kemungkinan besar memiliki tingkat penjualan yang lebih tinggi selama musin hujan dibandingkan dengan musim panas.

Pola *cycle* tidak berulang dalam waktu dekat. Biasanya dihaslikan dari faktor eksternal yang membuat pola ini lebih sulit untuk diprediksi. Metode prediksi biasanya memanfaatkan pola-pola tersebut untuk menghasilkan prediksi yang handal. Tren, *seasonality*, dan *cycle* merupakan pola yang paling umum ditemui dalam data *time series*. Mengetahui pola apa yang terdapat dalam data adalah hal penting. Tergantung pada metode apa yang dipilih, komponen tren, *seasonality*, dan *cycle* dapat diterapkan secara berbeda-beda.

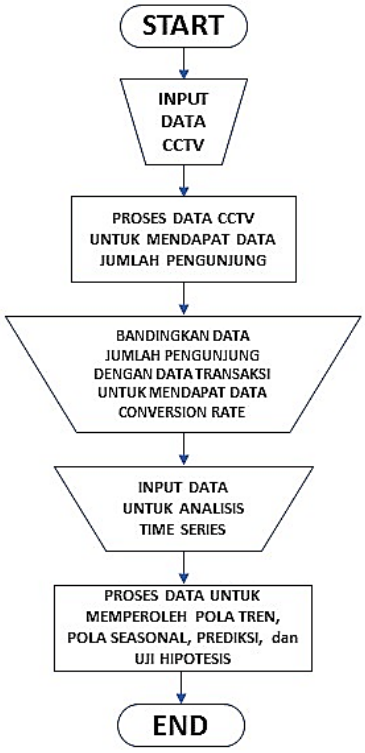
Untuk analisis *time series* perlu dilakukan penghitungan bobot agar tren, *seasonal*, dan *cycle* dapat dicari kombinasi terbaik dari ketiganya sehingga membentuk model yang mampu melakukan prediksi untuk waktu yang akan datang. *Time series analysis* dapat digunakan untuk melakukan prediksi jumlah pengunjung dengan menggunakan beberapa metode seperti *simple moving average*, *weighted moving average*, atau juga *ARIMA* untuk kalkulasi tren, berdasarkan data historis selama beberapa kurun waktu terakhir [20].

III. PERANCANGAN

Pada bagian ini, menjelaskan mengenai alur proses penelitian (*flowchart*) dan desain dari penelitian yang dilakukan. Desain penelitian membahas mengenai penerapan teori dan pengetahuan yang dimiliki terhadap data sehingga dapat menghasilkan *output* yang diharapkan dapat digunakan untuk pengambilan keputusan. Dalam penelitian ini yang menjadi objek penelitian adalah jumlah pengunjung toko *retail X* yang dihitung dengan melalui data *video cctv* dan data transaksi toko *retail X* pada periode waktu yang sama, yang kemudian akan dianalisis secara *time series*.

A. Alur Penelitian

Pada gambar 2, adalah rancangan *flowchart* yang digunakan dalam melakukan penelitian ini. Pertama-tama dimulai dengan menggunakan data *video cctv* yang diperoleh dari toko *retail X* sebagai *input* pada program penghitungan orang dengan tujuan untuk memperoleh data penghitungan pengunjung. Data penghitungan pengunjung yang diperoleh kemudian dibandingkan dengan data transaksi dalam kurun waktu yang sama untuk memperoleh data *retail conversion rate*. Kemudian data-data tersebut dianalisis dengan metode *time series ARIMA* untuk mengetahui gambaran pola tren, pola *seasonal*, prediksi pada waktu yang akan datang, dan menguji hipotesis.



Gambar 2. *Flowchart* penelitian

B. Desain Eksperimen

Penelitian ini menggunakan pendekatan empiris kuantitatif dalam pelaksanaannya. Pendekatan ini berfokus untuk memperoleh hasil analisis dari pola tren dan *seasonality* dari data penghitungan orang yang didapatkan dengan memanfaatkan teknologi *computer vision*. Analisis data yang digunakan adalah analisis *time series* dengan metode *ARIMA* dan *Holt-Winter*. Kedua metode ini dapat digunakan untuk membuat prediksi jumlah pengunjung di masa yang akan datang.

Ada pun hal yang dilakukan sebelum memulai analisis *time series* adalah memperoleh data jumlah pengunjung melalui rekaman *video cctv* yang disediakan oleh toko *retail X*. Dalam memperoleh data tersebut, digunakan tekonologi *computer vision* dengan *deep learning* untuk pendeteksian citra penghitungan pengunjung [21]. Penelitian ini menggunakan model *MobileNet SSD* (*Single-Shot multibox Detection*) untuk mendeteksi objek yang sudah dibentuk dan dilatih sebelumnya oleh chuanqi305. Model tersebut merupakan versi *Caffe* dari implementasi awal dengan *TensorFlow*. Model ini dilatih pada *COCO dataset* dan dilakukan *fine tuning* pada *PASCAL VOC* sehingga memperoleh 72,7% *mAP* (*mean average precision*).

Selain menggunakan model tersebut, dibutuhkan juga sejumlah *Python library*, seperti:

- *Numpy;*
- *OpenCV*
- *Dlib*
- *Imutils*

Data *video cctv* yang menjadi *input* untuk program penghitungan orang memiliki kurun waktu 3 minggu yang dimulai dari tanggal 24 September 2020 hingga 14 Oktober 2020. Data tersebut memiliki ukuran piksel sebesar 640 yang kemudian diperkecil menjadi 500 piksel untuk mempercepat proses penghitungan orang. Selain itu, format warna *BGR* data *video cctv* juga dirubah menjadi format warna *RGB* yang bisa dibaca oleh *Dlib* untuk *object tracking*.

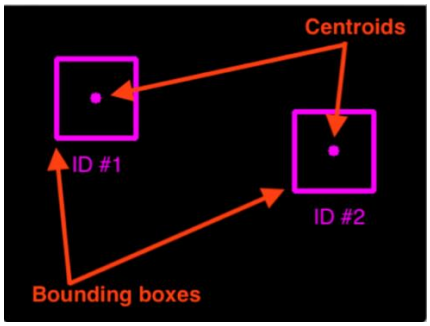
Model *MobileNet SSD* berfungsi untuk melakukan deteksi objek berupa orang, yang kemudian melacak objek tersebut selama dalam *frame*. Karena ini adalah model yang sudah dibentuk dan dilatih sebelumnya, dengan metode ini hanya memasukkan serta melewatkan *blob* (*wrapper* dari data yang sebenarnya akan diproses) ke dalam model tersebut [22]. Secara lebih lanjut, kedua fase penting, deteksi objek dan pelacakan objek, adalah sebagai berikut:

- *Detecting* – Selama fase ini, akan dijalankan *object detector* untuk mendeteksi apakah ada objek baru yang memasuki wilayah *frame* dan memberikan *bounding box* pada objek tersebut. Dalam studi ini, wilayah *frame* adalah wilayah tampilan *video cctv*. Selain itu, *object detector* juga melihat apakah objek sebelumnya yang ada pada *frame* sudah tidak ditemukan lagi (sudah keluar dari *frame*). Deteksi objek memerlukan daya komputasi yang cukup besar untuk bekerja. Maka dari itu, sebaiknya dibantu dengan *object tracker* yang meringankan kebutuhan akan daya komputasi [23].
- *Tracking* – Seperti yang disebutkan pada poin di atas, *object tracker* dapat membantu dalam meringankan kebutuhan akan daya komputasi dari *object detector*. Caranya adalah dengan melakukan proses deteksi sekali saja ketika setiap objek baru muncul dalam *frame*. Ketika *object detector* sudah berhasil melakukan pendeteksian objek dan memberikan *bounding box* pada objek, maka *object tracker* akan melakukan pelacakan pada objek selama masih berada dalam *frame* dengan memberikan ID dan titik *centroid* pada objek.

Perlu dilakukan cara untuk menyimpan informasi terkait dengan objek itu sendiri, yaitu:

- *Object ID;*
- Posisi *centroid* sebelumnya dalam *frame;*
- Menentukan apakah objek bersangkutan sudah pernah dihitung sebelumnya atau belum sama sekali.

Keuntungan yang bisa diperoleh dari metode ini adalah berkurangnya kebutuhan akan daya komputasi yang cukup besar karena proses deteksi objek tidak dijalankan pada setiap saat. Cara kerja deteksi dan pelacakan objek secara umum dapat dilihat pada gambar 3.



Gambar 3. Deteksi dan Pelacakan Objek

Setelah melakukan pendeteksian dan pelacakan objek berupa orang, untuk memperoleh jumlah orang yang ada dalam *frame* dan berjalan ke arah pintu masuk toko *retail* X, perlu dibentuk sebuah garis horizontal memanjang sebagai penanda apabila objek berupa orang melewati garis tersebut, maka jumlah penghitungan terhadap objek akan bertambah.

Dalam menghitung dan menentukan objek orang berjalan ke arah atas (*up*) atau ke bawa (*down*), perlu menuliskan kode yang memberitahu sistem agar dapat membedakan kedua hal tersebut. Cara kerja dari penghitungan orang dengan garis horizontal yang dibentuk dapat dilihat pada gambar 4.

Gambar 4. Penghitungan Orang

Pada saat data penghitungan orang sudah selesai didapatkan, kemudian data tersebut dibandingkan dengan data transaksi pada kurun waktu yang sama sehingga menghasilkan data *retail conversion rate*. Data transaksi adalah data yang diperoleh dari setiap nota penjualan tercetak. Setelah memperoleh hasil *retail conversion rate* toko *retail X*. Data jumlah pengunjung dan data transaksi menjadi *input* untuk analisis *time series*. Penelitian ini menggunakan *Jupyter Notebook* pada *Google Colab* dan *library Python*, seperti:

- *Numpy;*
- *Matplotlib;*
- *Pandas;*
- *Statsmodels.api.*

Analisis *time series* ini dilakukan untuk mencari tahu tentang bentuk pola tren, *seasonal*, hingga melakukan prediksi jumlah pengunjung dan jumlah transaksi di masa yang akan datang. Selain itu, analisis ini juga memungkinkan untuk memperoleh informasi mengenai kapan hari-hari di mana toko retail X dipadati oleh pengunjung dan hari-hari mana yang mencetak transaksi terbanyak, begitu pun sebaliknya.

Informasi ini bisa menjadi alasan bagi pelaku usaha retail untuk membuat keputusan, misalnya penempatan jumlah staff di dalam toko sehingga bisa meningkatkan potensi transaksi dan nilai retail convertion rate. Analisis ini juga dilakukan untuk mencari tahu apakah data jumlah pengunjung memiliki pola yang sama dengan data transaksi. Dalam melakukan analisis *time series*, teradapat tiga metode yang dilakukan dibandingkan pada studi ini, yaitu metode *Seasonal ARIMA (SARIMAX)*, *Exponential Smoothing Holt-Winter*, dan *FB Prophet.*

IV. IMPLEMENTASI DAN EKSPERIMEN

Pada bagian ini, menjelaskan mengenai penerapan dan hal-hal yang dilakukan berdasarkan dari desain penelitian yang dipaparkan pada bab sebelumnya. Secara garis besar, hal pertama yang dilakukan adalah memperoleh data jumlah pengunjung dari sistem penghitungan orang. Data jumlah pengunjung kemudian dibandingkan dengan data transaksi sehingga memperoleh data *retail conversion rate*. Data jumlah pengunjung dan data transaksi kemudian diolah secara *time series* untuk menunjukkan pola tren, *seasonality*, hingga melakukan prediksi di masa yang akan datang.

A. Penghitungan Pengunjung

Program penghitungan pengunjung ini digunakan untuk menggantikan fungsi dan juga tugas penghitungan manual yang sebelumnya dilakukan oleh *staff* dengan alat bantu berupa *clicker* manual. Berikut adalah langkah-langkah yang dilakukan.

1) Input data: Data yang digunakan sebagai *input* pada program penghitungan pengunjung ini adalah data rekaman *video CCTV* yang disediakan oleh toko *retail X*. Data ini dimasukkan ke dalam program penghitungan pengunjung dengan *parse* argumen seperti pada program 1.

```
# Buat argumen parser, dan parse argumen yang
diinginkan
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
   help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
   help="path to Caffe pre-trained model")
ap.add_argument("-i", "--input", type=str,
   help="path to optional input video file")
ap.add argumnent("-o", "--output", type=str,
   help="path to optional output video file")
ap.add_argument("-c", "--confidence", type=float,
default=0.4,
   help="probabilitas minimum untuk filter deteksi
lemah")
ap.add_argument("-s", "—skip-frames", type=int,
default=30,
   help="# jumlah frame yang dilewati antara
deteksi")
args = vars(ap.parse_args(args =
   ['--prototxt', 'C:/Users/Windows/people-
counting-
opencv/mobilenet_ssd/MobileNetSSD_deploy.prototxt'
,
    '--model', 'C:/Users/Windows/people-counting-
opencv/mobilenet_ssd/MobileNetSSD_deploy.caffemode
l',
    '--input',
'F:/Data_CCTV/FOLDER_CCTV/2020/hiv00028.mp4',
    '--ouput', 'F:/output/0925.avi']))
```

Program 1. *Input* data rekaman *video CCTV*

2) Resize piksel dan rubah format warna data: Data *video cctv* yang menjadi *input* untuk program penghitungan orang memiliki kurun waktu 3 minggu yang dimulai dari tanggal 24 September 2020 hingga 14 Oktober 2020. Data tersebut memiliki ukuran piksel sebesar 640 yang kemudian diperkecil menjadi 500 piksel untuk mempercepat proses penghitungan orang. Selain itu, format warna BGR data *video cctv* juga dirubah menjadi format warna RGB yang bisa dibaca oleh *Dlib* untuk *object tracking*. Kode untuk kedua proses tersebut dapat dilihat pada program 2.

```
# Memperkecil ukuran frame menjadi 500 piksel
#(semakin kecil data, semakin cepat prosesnya)
# Mengubah format warna BGR menjadi RGB untuk
Dlib
# Object Tracker
frame = imutils.resize(frame, width=500)
rgb = cv2.cvtColor (frame, cv2.COLOR_BGR2RGB)
```

Program 2. *Resize* piksel dan mengubah format warna

3) Penggunaan object tracker: Dalam program untuk menghitung jumlah pengunjung, model yang digunakan adalah *pre-trained model MobileNet SSD* seperti yang telah disebutkan pada bab sebelumnya. *Object detector* pada model tersebut dibantu dengan *object tracker* untuk meringankan kebutuhan akan daya komputasi. Kode *object tracker* pada program 3.

```
# Melacak dan menghitung object pada video
class TrackableObject:
   def __init__(self, objectID, centroid) :
      # menyimpan object ID, lalu inisialisasi
daftar centroid
      # menggunakan centroid saat ini
      self.objectID = objectID
      self.centroids = [centroid]

      # inisialisasi Boolean yang digunakan
untuk menunjukkan
      # apakah objek sudah dihitung atau belum
      self.counted = False
```

Program 3. Kode untuk *trackable object*

4) Membentuk garis virtual: Agar memudahkan penghitungan orang, perlu dibentuk garis horizontal seperti pada program 4 sebagai batas, apabila objek orang bergerak melewati garis tersebut dari posisi atas *frame* menuju ke bawah, maka jumlah penghitungan "ke bawah" akan bertambah, apabila objek orang bergerak melewati garis tersebut dari posisi bawah *frame* menuju ke atas, maka jumlah penghitungan "ke atas" akan bertambah.

```
# Membentuk garis horizontal untuk menghitung
objek orang
   # Garis dibentuk pada posisi center dari
frame
   # Objek yang melewati garis akan ditentukan
apakah bergerak ke
   # atas atau ke bawah
   cv2.line(frame, (0, H // 2), (W, H // 2), (0,
255, 255), 2)
```

```
    # gunakan pelacak centroid untuk mengaitkan
objek lama (1)
    # centroid dengan (2) adalah centroid objek
yang baru dihitung
    objects = ct.update(rects)
```

Program 4. Membuat garis horizontal

5) Menentukan objek naik atau turun: Dalam menghitung dan menentukan objek orang berjalan ke arah atas (*up*) atau ke bawah (*down*), perlu menuliskan kode seperti pada program 5 yang memberitahu sistem agar dapat membedakan kedua hal tersebut.

```
# Mengecek apakah objek sudah dihitung atau
belum
if not to.counted:
    # jika arah tujuan negative (indikasi objek
bergerak ke atas)
    # dan titik centroid berada di atas garis
horizontal, maka
    # hitung objek tersebut
    if direction < 0 and centroid[1] < H // 2:
        totalUp += 1
        to.counted = True

    # jika arah tujuan positif (indikasi objek
bergerak ke bawah)
    # dan titik centroid berada di bawah garis
horizontal, maka
    # hitung objek tersebut
    elif direction > 0 and centroid[1] > H // 2:
        totalDown += 1
        to.counted = True
```

Program 5. Menentukan arah objek naik atau turun

Dengan program penghitungan pengunjung, data *video CCTV* yang menjadi *input* diolah menjadi keluaran berupa *video* dengan jumlah penghitungan orang seperti pada gambar 5. Pada gambar sebelah kiri adalah *frame* sebelum diproses dengan penghitungan orang, sedangkan di sebelah kanan adalah gambar hasil proses penghitungan orang.

Gambar 5. Hasil penghitungan pengunjung (kanan)

B. Data Transaksi

Setelah melakukan penghitungan data pengunjung melalui pemanfaatan teknologi *computer vision*, langkah selanjutnya adalah memperoleh data terkait transaksi sehingga nantinya kedua data bisa dibandingkan untuk memperoleh *retail conversion rate*. Data transaksi diperoleh melalui pihak toko *retail X* dalam bentuk *file excel*. Data transaksi tersebut memiliki kurun waktu yang sama dengan kurun waktu data jumlah pengunjung, yaitu berkisar pada tanggal 24 September 2020 hingga 14 Oktober 2020.

Data transaksi berisikan seluruh rekaman penjualan toko *retail X* yang terjadi pada tanggal-tanggal tersebut. Setiap terjadi penjualan (nota tercetak) maka merupakan 1 transaksi. Setelah melakukan penghitungan pengunjung, data jumlah pengunjung yang diperoleh dibandingkan dengan data transaksi dalam kurun waktu yang sama. Tujuannya adalah untuk memperoleh data *retail conversion rate* seperti pada Tabel 1.

TABEL I
RETAIL CONVERSION RATE

Date	Visitor	Transaction	Conversion
24/09/2020	1910	2009	105.18
25/09/2020	2271	2325	102.38
26/09/2020	2958	3107	105.04
27/09/2020	3789	3911	103.22
28/09/2020	2045	2093	102.35
29/09/2020	2171	2212	101.89
30/09/2020	2224	2510	112.86
01/10/2020	2714	2874	105.90
02/10/2020	2525	2824	111.84
03/10/2020	4085	3885	95.10
04/10/2020	5139	5172	100.64
05/10/2020	2126	2458	115.62
06/10/2020	1935	2281	117.88
07/10/2020	2095	2422	115.61
08/10/2020	2253	2338	103.77
09/10/2020	2599	2721	104.69
10/10/2020	3511	3358	95.64
11/10/2020	4292	3917	91.26
12/10/2020	2233	2443	109.40
13/10/2020	1834	2143	116.85
14/10/2020	1965	1993	101.42

C. Analisis Time Series SARIMAX

Langkah berikutnya adalah melakukan analisis *time series* menggunakan data jumlah pengunjung dan data transaksi sebagai *input*. Analisis *time series* ini dilakukan untuk mencari tahu tentang bentuk pola tren, *seasonal*, hingga melakukan prediksi jumlah pengunjung dan jumlah transaksi di masa yang akan datang. Selain itu, analisis ini juga memungkinkan untuk memperoleh informasi mengenai kapan hari-hari di mana toko *retail X* dipadati oleh pengunjung dan hari-hari mana yang mencetak transaksi terbanyak, begitu pun sebaliknya. Informasi ini bisa menjadi *insight* bagi pelaku usaha *retail* untuk membuat keputusan, misalnya penempatan jumlah *staff* di dalam toko sehingga bisa meningkatkan potensi transaksi dan nilai *retail convertion rate.* (contoh yang digunakan: data *visitor*)

1) Preprocessing data: Sebelum melakukan analisis, data yang terdiri atas tiga kolom (*Date*, *Total*, dan *Category*) seperti pada Tabel 2, harus dijadikan *dataframe* terlebih dahulu.

TABEL II
DATAFRAME DENGAN HEAD (6)

	Date	Total	Category
0	2020-09-24	1910	Visitor
1	2020-09-24	2009	Transaction
2	2020-09-25	2271	Visitor

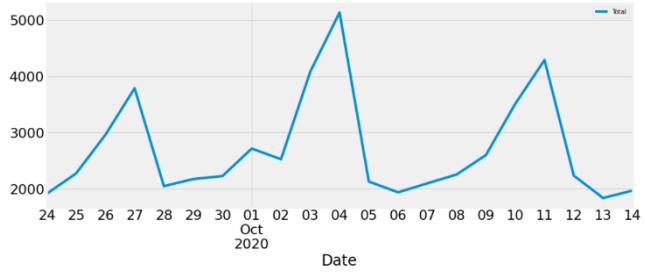
	Date	Total	Category
3	2020-09-25	2325	Transaction
4	2020-09-26	2958	Visitor
5	2020-09-26	3107	Transaction

Data tersebut harus dipisah berdasarkan *category* yang ada, yaitu *Visitor* dan *Transaction*. Tahap pertama adalah melakukan *preprocessing* terhadap terlebih dahulu. Tujuannya adalah untuk memilih data, baik data dengan kategori *Visitor* maupun *Transaction* dari *dataframe*. Kemudian memberikan *timestamp* yang menunjukkan kurun waktu data. Langkah penting berikutnya adalah melakukan *indexing* dan *resample* data agar data siap diproses sesuai dengan frekuensi waktu yang diinginkan (harian / *daily*) seperti pada program 6.

```
# Timestamp Data Visitor
# Untuk mengetahui kurun waktu dari data
Visitor['Date'].min(), Visitor['Date'].max()

# Output
(Timestamp('2020-09-24 00:00:00'),
Timestamp('2020-10-14 00:00:00'))

# Indexing Data Visitor
Visitor = Visitor.set.index('Date')
Visitor.index

# Memastikan frekuensi data Visitor dalam harian
V = Visitor['Total'].resample(rule='D').mean()
```

Program 6. *Processing* data

2) Visualisasi data: Visualisasi data *visitor* pada Gambar 6 menunjukkan gambaran adanya kenaikan dan penurunan jumlah pengunjung yang datang ke toko *retail X* selama masa waktu observasi. Jika dilihat, pada setiap menjelang akhir pekan, dimulai dari hari Jumat (tanggal 25, 2, dan 9), jumlah pengunjung mengalami kenaikan hingga puncaknya pada hari Minggu (tanggal 27, 4, dan 11). Setelahnya, jumlah pengunjung yang datang ke toko retail X mengalami penurunan pada hari Senin (tanggal 28, 5, dan 12). Dengan adanya gambaran yang menunjukkan kenaikan dan penurunan yang terjadi pada setiap minggu (waktu yang sama), dapat dikatakan bahwa dalam data tersebut terdapat *seasonality*.



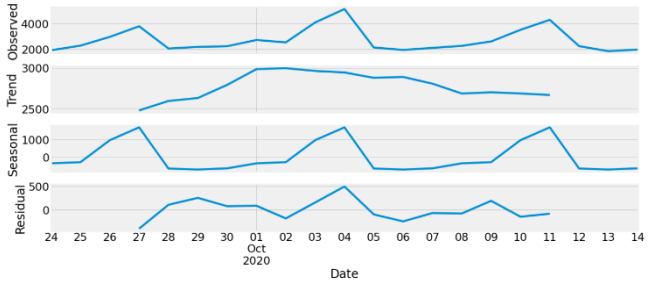
Gambar 6. Visualisasi data *visitor*

3) Dekomposisi data: Perlu melakukan dekomposisi agar pola tren dan pola *seasonal* dapat terlihat tanpa adanya *noise / residual* seperti pada program 7.

```
# Dekomposisi Time Series
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition =
sm.tsa.seasonal_decompose(Visitor,
model='additive')
fig = decomposition.plot()
plt.show()
```

Program 7. Dekomposisi data *time series*

Pola tren dan *seasonal* dapat divisualisasikan secara terpisah dan ditampilkan secara bersamaan dengan visualisasi data sebelum dekomposisi (*Observed*) seperti pada gambar 7 Terlihat bahwa pola tren mengalami kenaikan di awal, lalu berubah menjadi menurun pada akhir. Pola *seasonal* terbentuk dengan baik, di mana pada setiap kurun waktu tertentu mengalami kenaikan yang sama dan sifatnya berulang.



Gambar 7. Hasil dekomposisi data *visitor*

4) Mencari parameter terbaik ARIMA data: Untuk menemukan nilai-nilai terbaik dari parameter *ARIMA*, dapat dilakukan dengan menggunakan *grid search* (optimisasi *hyperparameter* untuk memilih model) yang menjelajah dan mencari berbagai kombinasi parameter secara berulang seperti pada program 8.

```
# ARIMA Time Series Model
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 7) for x in
list(itertools.product(p, d, q))]
print('Examples of parameter combinations for
Seasonal ARIMA…')
print('SARIMAX: {} x {}'.format(pdq[1],
seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1],
seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2],
seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2],
seasonal_pdq[4]))
```

Program 8. Kombinasi parameter *Seasonal ARIMA*

Dengan kode tersebut, dapat menggunakan *triplet* parameter yang ditentukan untuk mengotomatiskan proses pelatihan dan mengevaluasi model *ARIMA* pada kombinasi yang berbeda. Saat mengevaluasi dan membandingkan model yang dilengkapi dengan parameter berbeda, masing-masing dapat diberikan peringkat berdasarkan seberapa cocok model tersebut dengan data atau kemampuannya untuk secara akurat melakukan prediksi titik data di masa depan. Peringkat diurutkan berdasarkan nilai *AIC* (*Akaike Information Criterion*), yang

p-ISSN : 2443-2210
e-ISSN : 2443-2229

dengan mudah dikembalikan dengan model *ARIMA* menggunakan *statsmodels*.

AIC mengukur seberapa cocok suatu model dengan data sambil mempertimbangkan kompleksitas model secara keseluruhan. Model yang sangat cocok dengan data saat menggunakan banyak fitur akan diberi nilai *AIC* yang lebih besar daripada model yang menggunakan lebih sedikit fitur untuk mencapai kesesuaian yang sama. Oleh karena itu, model yang menghasilkan nilai *AIC* terendah merupakan model yang dipilih [24]. Model dengan *AIC* terendah dapat dicari seperti pada program 9.

```
# MENCARI KOMBINASI PARAMETER TERBAIK UNTUK MODEL
for param in pdq:
  for param_seasonal in seasonal_pdq:
    try:
      mod = sm.tsa.statespace.SARIMAX(V,

order=param,

seasonal_order=param_seasonal,

enforce_stationarity=True,

enforce_invertibility=True)
      result = mod.fit()
      print('ARIMA{}x{}7) -
AIC:{}'.format(param, param_seasonal,
results.aic))
    except:
      continue
```

Program 9. Mencari model dengan parameter terbaik

5) Fitting ARIMA model: Setelah melakukan *run* pada kode program tersebut, muncul hasil berupa model dengan kombinasi parameter yang sangat banyak, akan tetapi, model yang perlu dilihat adalah yang terdapat pada urutan terakhir, karena model itulah yang memiliki nilai *AIC* terendah, yaitu *ARIMA*(1, 1, 1)x(1, 1, 0, 7)7 – *AIC*:207.1063. Model tersebut memiliki nilai *AIC* terkecil dibandingkan dengan model lainnya sehingga model tersebut dianggap sebagai model terbaik dan optimal. Langkah berikutnya, menganalisis model dengan parameter yang sudah diperoleh tadi seperti pada program 10.

```
#Fitting Model
mod = sm.tsa.statespace.SARIMAX(V,
                                order=(1, 1, 1),

seasonal_order=(1, 1, 0, 7),

enforce_stationarity=True,

enforce_invertibility=True)
results = mod.fit()
print(results.summary().tables[1])
```

Program 10. *Fitting* model

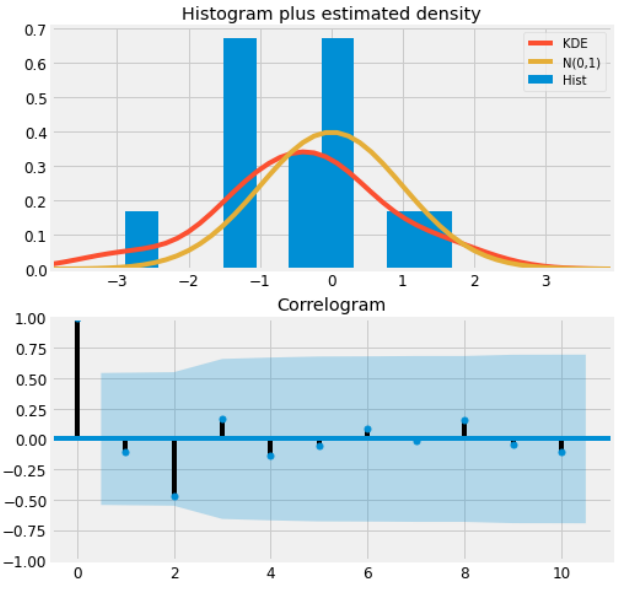
Ringkasan atribut yang dihasilkan dari keluaran *SARIMAX* mengembalikan sejumlah besar informasi pada gambar 8. Kolom *coef* menunjukkan bobot (tingkat kepentingan) setiap fitur dan bagaimana masing-masing fitur mempengaruhi *time series*. Kolom *P*>|*z*| memberitahu tentang pentingnya setiap bobot fitur. Di sini, bobot dengan nilai p yang mendekati atau di bawah *significance level* α = 5% *(default)* dianggap lebih penting dibandingkan bobot fitur lainnya.

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2949	1.335	0.221	0.825	-2.322	2.912
ma.L1	-0.5533	1.598	-0.346	0.729	-3.686	2.580
ar.S.L7	-0.7042	0.446	-1.578	0.115	-1.579	0.171
sigma2	1.325e+05	8.57e+04	1.545	0.122	-3.56e+04	3.01e+05

Gambar 8. *Summary* model *visitor*

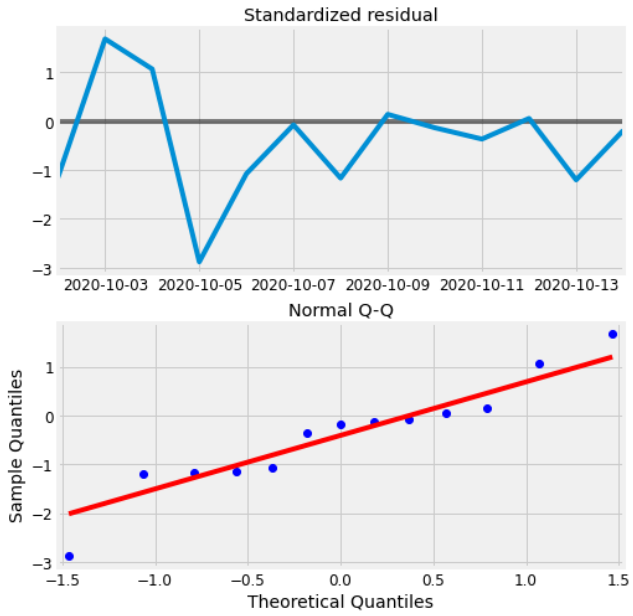
Terlihat bahwa nilai *P*>|*z*| dari setiap atribut berada di atas dari *significance level* α = 5%. Hal ini bisa saja terjadi karena data observasi kurang banyak. Prediksi masih dapat dilakukan oleh model, hanya saja mungkin dikarenakan waktu observasi yang kurang panjang, prediksi yang dilakukan memiliki rentang perubahan hasil yang cukup besar.

6) Diagnostik model: Diagnostik model memberikan keluaran yang dapat dianalisis untuk menyimpulkan apakah residual pada model berdistribusi normal. Pada gambar 9 *Histogram* tidak tersusun dengan baik karena data poin terbilang sedikit hanya saja garis *KDE* (*Kernel Density Estimation*) berwarna merah memiliki bentuk alur yang menyerupai dan berdekatan dengan garis berwarna kuning N(0, 1) serta plot *correlogram* menunjukkan tidak adanya korelasi data *time series* dengan versi lagnya terdahulu, ini mengindikasikan bahwa residual berdistribusi normal.



Gambar 9. Histogram dan Correlogram model visitor

Selain itu, pada gambar 10 menunjukkan residual dari waktu ke waktu tidak menyimpulkan suatu pola *seasonal* tertentu hanya saja terlihat berfluktuasi bukan pada *mean* 0. Pada Plot Q-Q (*quantile-quantile*) diperlihatkan bahwa sebaran distribusi residu sudah teratur mengikuti tren linier dari sampel yang diambil dari distribusi normal standar dengan N(0, 1). Ini merupakan indikasi bahwa residu dari model berdistribusi normal.



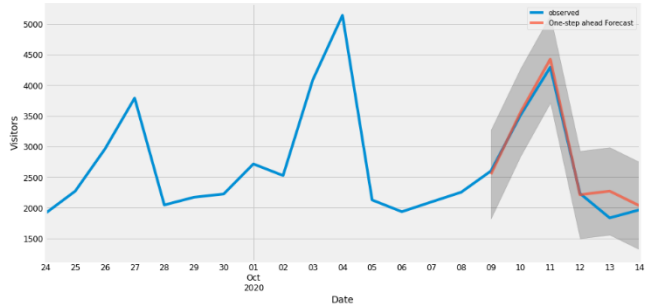
Gambar 1**0**. Plot Residual dan Plot QQ model data *visitor*

7) Validasi model: Setelah melakukan diagnostik model, langkah berikutnya adalah melakukan validasi prediksi seperti pada program 11 terhadap model yang dibentuk. Dengan membandingkan antara jumlah pengunjung yang diprediksi dengan jumlah pengunjung sebenarnya (*observed*), untuk mengetahui apakah model yang dibentuk dapat melakukan prediksi dengan baik.

```
# Validasi Prediksi Visitor
pred =
results.get_prediction(start=pd.to_datetime('2020-
10-09'), dynamic=False)
pred_ci = pred.conf_int()
ax = V['2020':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label="One-step
ahead Forecast", alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 0], color='k',
alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('Visitors')
plt.legend()
plt.show()
```

Program 11. Validasi prediksi *visitor*

Hasil prediksi pada gambar 11 menunjukkan bahwa jumlah pengunjung yang diprediksi (warna orange) mengikuti dan mendekati jumlah pengunjung sebenarnya (*observed*, warna biru). Jumlah pengunjung diprediksi ditentukan pada tanggal 9 Oktober 2020 hingga masa waktu data *observed* habis (14 Oktober 2020). Selama masa waktu tersebut, pola yang dibentuk oleh prediksi cukup mendekati data sebenarnya.



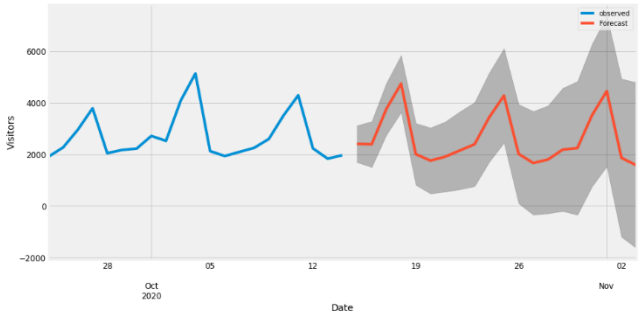
Gambar 11. Hasil validasi prediksi

Hal serupa dapat dilakukan untuk mengetahui prediksi yang menggambarkan *seasonality* jumlah pengunjung toko *retail X* di masa depan seperti pada program 12. Semakin jauh prediksi yang dilakukan, maka *margin error* yang dihasilkan oleh model semakin besar Hal tersebut adalah wajar mengingat rentang waktu data observasi yang digunakan kurang banyak.

```
pred_uc = results.get_forecast
(steps=20)
pred_ci = pred_uc.conf_int()
ax = V['2020':].plot(label='observed',
figsize=(14, 7))
pred_uc.predicted_mean.plot(ax=ax,
label='Forecast')

ax.fill between (pred ci.index,

                pred_ci.iloc[:, 0],

                pred_ci.iloc[:, 1], color='k', alp

ha=.25)

ax.set_xlabel('Date')

ax.set_ylabel('Visitors')

plt.legend()
plt.show()
```

Program 12. Prediksi data dengan *seasonality* masa depan *visitor*

Pada gambar 12 terlihat dari garis abu-abu di sekitar garis orange yang mewakili prediksi *seasonality* jumlah pengunjung. Namun, hasil prediksi *seasonality* yang dihasilkan tetap memiliki pola kenaikan dan penurunan yang serupa dengan *seasonality* data sebenarnya. Apabila rentang waktu data observasi semakin panjang, maka interval kepercayaan akan semakin kuat pula (garis abu-abu semakin mengecil).



Gambar 12. Visualisasi prediksi *seasonality* data

8) Akurasi model: tingkat akurasi prediksi dari model yang telah dibentuk dapat diketahui dengan melihat skor *RMSE*

p-ISSN : 2443-2210
e-ISSN : 2443-2229

(*Root Mean Squared Error*). Model *SARIMAX* data jumlah pengunjung memiliki nilai *RMSE* sebesar 287.25.

9) Data Transaksi: Sama halnya dengan data pengunjung yang dianalisis secara *time series* dengan metode *SARIMAX*, data transaksi juga mengikuti langkah-langkah analisis yang telah disampaikan pada poin-poin di atas. Model prediksi *SARIMAX* data transaksi memiliki skor *RMSE* sebesar 351.57.

D. Analisis Time Series Holt-Winter

Metode berikutnya yang akan digunakan adalah metode *exponential smoothing* atau lebih dikenal dengan nama *Holt-Winter Method*. Metode ini melakukan optimalisasi terhadap parameter *level*, tren, dan *seasonal* dari pembentukan model dengan memberikan bobot masing-masing terhadap parameter-parameter tersebut. Selain itu, dalam metode ini, kriteria penilaian yang digunakan adalah *AICc* (Corrected *Akaike Information Criterion*). Kriteria penilaian ini adalah kelanjutan dari *AIC* yang sebelumnya digunakan pada metode *SARIMAX*. Kriteria *AICc* digunakan karena jumlah sampel data yang sedikit (n<30). Berikut adalah langkah-langkah yang dilakukan (menggunakan contoh data *visitor*).

1) Train test split data: Langkah awal yang dilakukan adalah menentukan dan memisahkan data jumlah pengunjung untuk pelatihan dan pengujian. Disarankan untuk data yang dijadikan untuk pelatihan terdapat setidaknya 2 seasonal di dalamnya [25]. Train dan split data jumlah pengunjung dapat dilakukan seperti pada program 13. Hasil yang diperoleh melalui program ini adalah parameter BoxCox sebesar -1.81 untuk melakukan transformasi data deret waktu sehingga mencapai linearitas atau memiliki varians yang konstan.

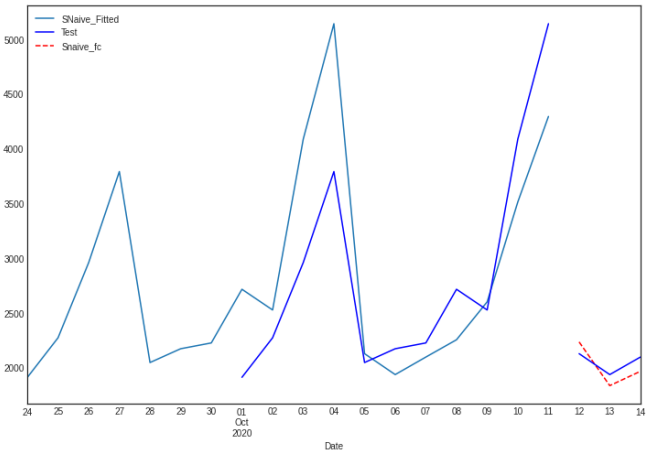
```
# Split into train and test
train = Visitor.iloc[:-3]
test  = Visitor.iloc[:-3]
# Forecast horizon
h = 3
train_length = len(train)

print('train_length:',train_length,'\ntest_length'
,len(test))

# Creating BxCx transformed train & test to be
used later
train_bcox, bcox_lam = boxcox(train["Total"])
print("BoxCox parameter to linearize the series:",
bcox_lam.round(2))
test_bcox = boxcox(test["Total"], lmbda=bcox_lam)

train_log = np.log(train["Total"])
```

Program 13. *Split* data *visitor* untuk *train* dan *test*

2) Seasonal Naive Forecast Model: Langkah berikutnya adalah melakukan peramalan dengan metode seasonal naive yang menggunakan pengamatan dari musim (season) yang sesuai dari periode sebelumnya. Misalnya, perkiraan jumlah pengunjung pada hari Senin tanggal 12 adalah berdasarkan hari Senin pada periode sebelumnya. Hal ini tidak memperhitungkan tren sebelumnya. Metode ini bukanlah yang paling akurat, namun dapat menjadi dasar perbandingan bagi metode yang lebih kompleks. Metode ini dapat dilakukan seperti pada program 14. Hasil prediksi dari model dapat divisualisasikan untuk melihat apakah mengikuti atau mendekati alur data sebenarnya seperti pada gambar 13.

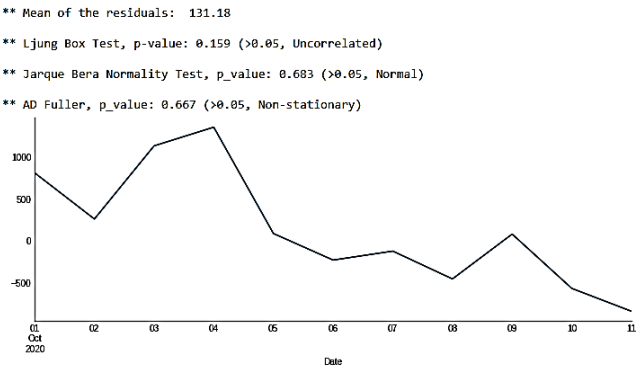
```
# Fitted values
py_snaive_fit = pysnaive(train["Total"],
                    seasonal_periods=7,
                    forecast_horizon=3)[0]

# Forecast
py_snaive = pysnaive(train["Total"],
                    seasonal_periods=7,
                    forecast_horizon=3)[1]

#Residuals
py_snaive_resid = (train["Total"] -
py_snaive_fit).dropna()

predictions["py_snaive"] = py_snaive.values

predictions
```

Program 14. *Seasonal naïve forecast visitor*



Gambar 13. Visualisasi *seasonal naive Forecast*

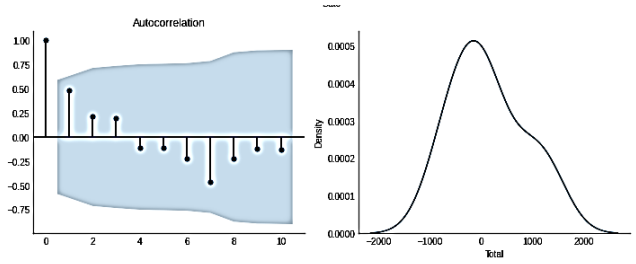
3) Residual Seasonal Naive Forecast: Langkah berikutnya yang dilakukan adalah melihat residual yang ada. Pada gambar 14 terlihat bahwa model yang dibentuk memiliki residu dengan *non-zero mean*.



Gambar 14. Distribusi residual model

Dari haril yang diperoleh melalui visualisasi gambar 15, terlihat bahwa model yang dibentuk memiliki performa cukup

baik. Analisis residual menunjukkan bahwa residu berdistribusi normal dan tidak stasioner. Plot autokorelasi memperlihatkan bahwa tidak ada lag yang signifikan (tidak berkorelasi).



Gambar 15. Plot autokorelasi dan histogram model *seasonal naive visitor*

4) Holt-Winter Method: Metode ini menguraikan data deret waktu menjadi level, tren, dan seasonal. Nilai masa depan diprediksi dengan menggabungkan faktor-faktor sistematis tersebut berdasarkan data historis terkini. Ide dari metode ini adalah bahwa masa depan akan berperilaku sangat mirip dengan masa lalu terkini. Langkah pertama yang dilakukan pada metode ini adalah melakukan grid searching dseperti pada program 15.

```
# Grid searching
m=HWGrid(train[“Total”], test[“Total”],
seasonal_periods=7)
m
```

Program 15. *Grid searching visitor*

Grid searching mengembalikan *dataframe* dengan parameter metode *Holt-Winter* dan skor evaluasi pelatihan serta pengujian yang sesuai. *Grid searching* juga melakukan pemeriksaan cepat terhadap residual menggunakan uji *Ljung-Box* dan uji *Saphiro* untuk normalitas seperti pada gambar 16.

	Trend	Seasonal	Damped	BoxCox	AICc Train	%MAPE_Train	RMSE_Train	%MAPE_Test	RMSE_Test	Resid_LJ	Resid_Norm	Resid_mean
5	add	add	True	log	353.500000	6.29%	193.700000	7.75%	159.300000	Uncorrelated	Normal	-26.300000
11	add	mul	True	log	351.900000	6.05%	184.800000	7.82%	160.200000	Uncorrelated	Normal	-23.500000
14	mul	add	False	log	303.700000	6.54%	199.800000	8.84%	176.300000	Uncorrelated	Normal	2.100000
20	mul	mul	False	log	302.300000	6.37%	192.200000	8.90%	177.600000	Uncorrelated	Normal	1.600000
2	add	add	False	log	303.700000	6.53%	199.700000	8.91%	177.800000	Uncorrelated	Normal	0.900000
8	add	mul	False	log	302.300000	6.36%	192.100000	8.97%	178.800000	Uncorrelated	Normal	0.600000
3	add	add	True	False	373.700000	11.21%	338.800000	9.31%	212.700000	Uncorrelated	Normal	-21.600000
0	add	add	False	False	323.800000	10.83%	349.100000	10.53%	226.900000	Uncorrelated	Normal	-22.900000
12	mul	add	False	False	322.300000	10.80%	335.200000	12.45%	248.500000	Uncorrelated	Normal	9.800000
15	mul	add	True	False	373.000000	11.42%	332.600000	13.26%	264.800000	Uncorrelated	Normal	-53.300000
23	mul	mul	True	log	351.800000	6.05%	184.500000	12.69%	284.600000	Uncorrelated	Normal	-23.700000
4	add	add	True	True	380.600000	9.74%	410.300000	13.15%	293.600000	Uncorrelated	Normal	-149.400000
10	add	mul	True	True	380.600000	9.74%	410.300000	13.15%	293.600000	Uncorrelated	Normal	-149.400000
1	add	add	False	True	329.500000	9.51%	409.500000	13.68%	299.800000	Uncorrelated	Normal	-152.000000
7	add	mul	False	True	329.500000	9.51%	409.500000	13.68%	299.800000	Uncorrelated	Normal	-152.000000
13	mul	add	False	True	329.500000	9.51%	409.500000	13.68%	299.800000	Uncorrelated	Normal	-152.000000
19	mul	mul	False	True	329.500000	9.51%	409.500000	13.68%	299.800000	Uncorrelated	Normal	-152.000000
9	add	mul	True	False	366.700000	9.21%	276.600000	16.24%	341.400000	Uncorrelated	Normal	-103.700000
21	mul	mul	True	False	366.200000	9.10%	275.000000	16.82%	356.400000	Uncorrelated	Normal	-94.600000
6	add	mul	False	False	315.900000	9.18%	280.600000	17.08%	362.800000	Uncorrelated	Normal	-109.300000
18	mul	mul	False	False	318.600000	9.96%	302.600000	19.32%	427.300000	Uncorrelated	Normal	-141.700000
17	mul	add	True	log	487.000000	266.57%	7869.100000	2,055,210.23%	41278645.000000	Uncorrelated	Normal	7523.000000
22	mul	mul	True	True	380.600000	9.74%	410.300000	nan%	nan	Uncorrelated	Normal	-149.400000
16	mul	add	True	True	nan	nan%	nan	nan%	nan	Correlated	Non-Normal	nan

Gambar 16. Hasil *grid searching*

5) Smoothing parameter: Dari *dataframe* hasil *grid search*, dapat dilihat bahwa *RMSE_Train* sebagian besar memiliki nilai yang lebih tinggi dibandingkan dengan *RMSE_Test*, ini menandakan bahwa hampir seluruh model yang dibentuk memiliki performa yang lebih baik pada set data pengujian. Semua model kecuali satu di bagian bawah memiliki residual yang tidak berkorelasi, ini menunjukkan bahwa semua model telah menangkap informasi sebanyak mungkin dari data yang tersedia. Semua model memiliki residual yang normal, kondisi ini dapat mempermudah penghitungan interval prediksi. Model yang dipilih adalah model yang sederhana dan parsimoni, oleh karena itu model harus memiliki *seasonality additive* serta yang memiliki skor *AICc* (*Corrected Akaike Information Criterion*) terbilang kecil yaitu 303.7. (model ke-5). Kemudian, model yang dipilih akan dilakukan *smoothing* seperti pada program 16.

```
hw_model = ExponentialSmoothing(train[“Sales”],
                trend      = “add”,
                seasonal   = “add”,
seasonal_periods = 7,
                damped =
False).fit(use_boxcox=’log’)

hw_fitted = hw_model.fittedvalues

hw_resid = hw_model.resid

# Adding the mean of the residuals to correct
the bias.
py_hw = hw_model.forecast(len(test[“Total”])) +
np.mean(hw_resid)

predictions[“py_hw”] = py_hw

# Holt-Winter Parameters
hw_model.params_formatted
```

Program 16. *Exponential smoothing*

Proses *smoothing* adalah optimalisasi parameter dengan cara memberikan bobot pada parameter *alpha*, *beta*, dan *gamma*. Pada gambar 17. menunjukkan bahwa paramater *alpha* dengan skor 0,7 yang artinya 70% bobot telah diberikan kepada observasi terakhir. Parameter *beta* adalah parameter pembelajaran bagi tren, sangat kecil skornya, tidak banyak bobot yang diberikan pada tren terkini dan tren yang diperoleh dari masa lalu. Parameter *gamma*, faktor *seasonal* sangat kecil, lebih kecil dari 0,2. Jika nilainya tinggi, maka akan memiliki risiko *overfitting* karena berarti model yang dibentuk terlalu banyak mempelajari data terkini. Skor yang kecil menandakan *seasonality* dipelajari dari observasi paling awal.

	name	param	optimized
smoothing_level	alpha	7.011388e-01	True
smoothing_trend	beta	6.628242e-18	True
smoothing_seasonal	gamma	5.847830e-16	True

Gambar 17. Hasil *exponential smoothing*

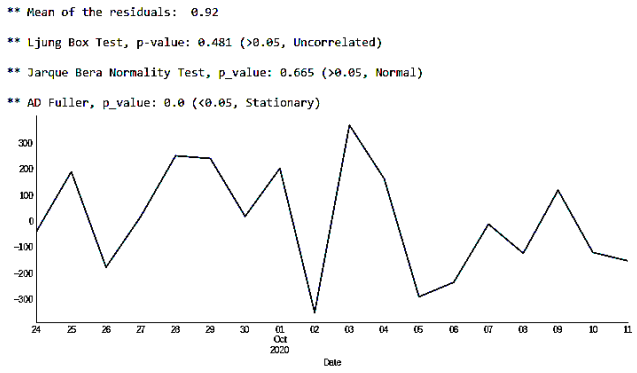
6) Cross Validation: Langkah berikutnya adalah melakukan *cross validation*. Dalam *girdsearch*, ukuran set pelatihan adalah tetap dan performa model dievaluasi dengan membandingkan skor *AICc*, *RMSE*, *%MAPE*. Metrik pengujian memberikan keakuratan perkiraan yang sebenarnya dan harus selalu digunakan untuk pemilihan model. Ini adalah pendekatan yang umum dilakukan pada saat ukuran data adalah besar. Namun, apabila data deret waktunya pendek, *cross validation* harus digunakan untuk memastikan model tidak *overfit*. Dalam melakukan *cross validation*, data yang digunakan sebagai pelatihan secara minimum harus memiliki dua buah *seasonal*. *Cross validation* dapat dilakukan seperti pada program 17.

p-ISSN : 2443-2210
e-ISSN : 2443-2229

```
hw_cv(Visitor["Total"], seasonal_periods=7,
  initial_train_window=18, test_window=3)
```

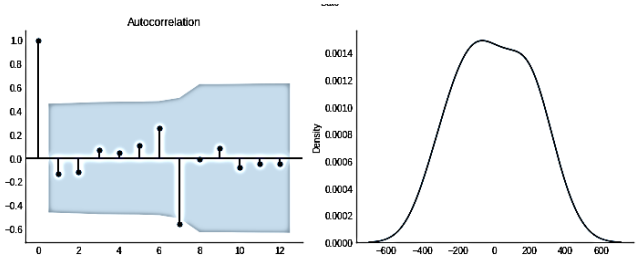
Program 17. *Cross validation visitor*

7) Residual check: Sama halnya dengan metode *seasonal naive* yang telah dilakukan sebelumnya, langkah selanjutnya adalah memeriksa residual dari model yang dibentuk. Dari pemeriksaan residual pada gambar 18. dapat dikatakan bahwa plot menunjukkan model yang dibentuk mampu menangkap tren, *seasonality* dengan baik. *Mean* pada residual berada di kisaran 0.



Gambar 18. Distribusi residual model *Holt Winter visitor*

Gambar 19 menunjukkan bahwa residual tidak berkorelasi, berdistribusi normal, dan stasioner. Model yang dibentuk dengan metode *Holt Winter* memiliki performa lebih baik dibandingkan dengan model yang dibentuk dengan metode *seasonal naive* (tanpa optimalisasi / pembobotan) ataupun dengan model yang dibentuk dengan *package statsmodel* (*SARIMAX* dengan skor *AIC*). Nilai %*MAPE* yang diperoleh adalah sebesar 8.9 dengan *RMSE* sebesar 178.1.



Gambar 19. Plot autokorelasi dan *histogram* model *Holt Winter*

8) Data transaksi: Sama halnya dengan data pengunjung yang dianalisis secara *time series* dengan metode *Holt-Winter*, data transaksi juga mengikuti langkah-langkah analisis yang telah disampaikan pada poin-poin di atas. Model prediksi *Holt-Winter* data transaksi memiliki nilai *%MAPE* sebesar 11.6% dan nilai *RMSE* sebesar 292.6.

E. Analisis Time Series FB Prophet

Metode ketiga dan terakhir yang digunakan adalah *FB Prophet*. Metode ini merupakan salah satu metode analisis *time series* terbaru yang dikembangkan oleh *Facebook* dan diluncurkan sebagai *open source* pada tahun 2017. Dalam metode ini, terdapat tiga komponen utama, yaitu tren, *seasonal*, *holidays* dan *error*. Langkah pertama yang harus dilakukan adalah (menggunakan contoh data *visitor*):

1) Mengecek tipe data: Hal terpenting adalah data pada kolom informasi waktu harus berjenis *datetime*, maka dari itu harus melakukan pengecekan seperti pada program 18.

```
Visitor.dtypes

# Output
Date          datetime64[ns]
Total                  int64
Category              object
dtype: object
```

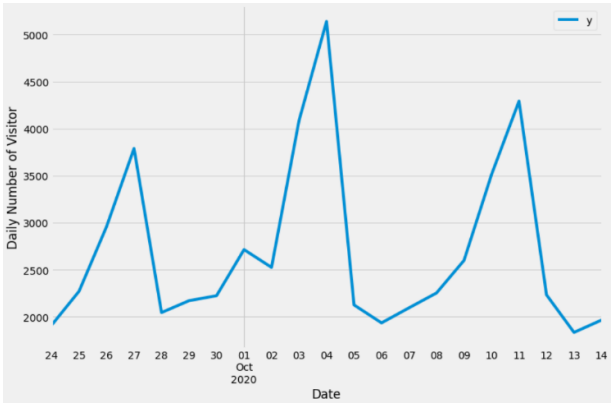
Program 18. Cek tipe data

2) Merubah kolom data: *Prophet library* mengharuskan *input* berupa *dataframe* yang memiliki satu kolom berisikan informasi waktu (ds) dan sebuah kolom lainnya berisikan metrik / nilai yang ingin diprediksi (y). Hal ini dapat dilakukan dengan program 19.

```
Visitor = Visitor.rename(columns=
   {'Date': 'ds', 'Total': 'y'})
```

Program 19. Merubah kolom data

3) Visualisasi data: sebelum melakukan analisis *time series* dengan *Prophet*, perlu dilakukan visualisasi data terlebih dahulu untuk mengetahui bentuk pergerakan data seiring waktu seperti pada gambar 20.



Gambar 20. Visualisasi persiapan *prophet* data *visitor*

4) Persiapan prediksi data: Sebelum melakukan prediksi data, perlu mengatur parameter *uncertainty interval* seperti pada program 20. *Uncertainty interval* mengembalikan nilai prediksi dengan ketidakpastian pada komponen tren dan *noise* dari data dengan asumsi rata-rata nilai perubahan dalam tren di masa yang akan datang memiliki kesamaan dengan data hasil observasi.

```
my_model = Prophet(interval_width=0.95)

my_model.fit(Visitor)
# Output
<fbprophet.forecaster.Prophet at 0x7f92d6f29750>
```

Program 20. *Uncertainty interval*

5) Prediksi dengan Prophet: Langkah berikutnya yang dilakukan adalah melakukan prediksi dengan periode

datestamp (ds) sebanyak 21. Penentuan jumlah periode *datsetamp* untuk prediksi dilakukan berdasarkan data observasi selama 21 hari. Apakah selama 21 hari kedepan, pola kenaikan dan penurunan memiliki kesamaan dengan data observasi? Dalam melakukan prediksi, perlu dipastikan juga frekuensi waktu yang digunakan sesuai dengan data, dalam kasus ini menggunakan frekuensi harian seperti pada program 21.

```
Future_dates =
my_model.make_future_dataframe(preiods=21,
freq='D')

forecast = my_model.predict(future_dates)
forecast[['ds', 'yhat', 'yhat_lower',
'yhat_upper']].tail()
```

Program 21. Prediksi dengan *prophet*

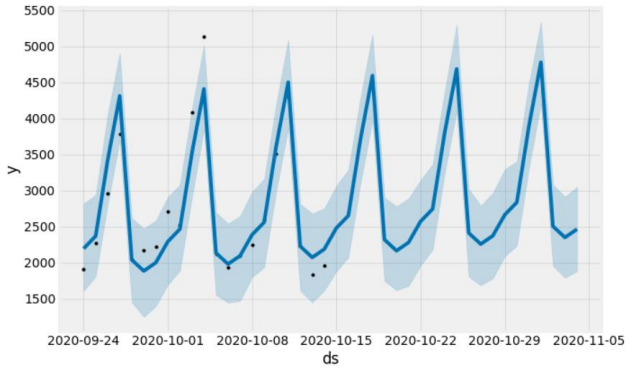
6) Dataframe baru hasil dari prediksi data: Setelah melakukan prediksi, terbentuk sebuah *dataframe* baru seperti pada Tabel III. *Dataframe* tersebut berisikan nilai-nilai hasil prediksi: *ds* (*datestamp* masa yang akan datang), *yhat* (nilai hasil prediksi), *yhat_lower* (batas bawah dari prediksi), *yhat_upper* (batas atas dari prediksi).

TABEL III

DATAFRAME BARU HASIL PREDIKSI

	ds	yhat	yhat_lower	yhat_upper
37	2020-10-31	3888.94	3307.68	4482.80
38	2020-11-01	4777.30	4206.68	5335.33
39	2020-11-02	2504.95	1956.09	3083.18
40	2020-11-03	2350.01	1791.21	2912.92
41	2020-11-04	2464.41	1879.83	3052.56

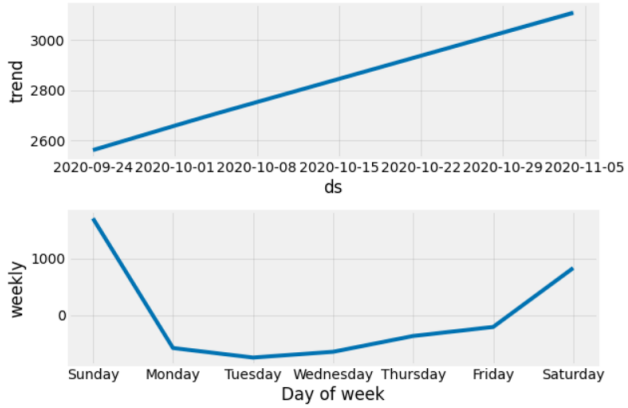
7) Visualisasi hasil prediksi: Setelah melakukan prediksi, Prediksi tersebut dapat divisualisasikan seperti pada gambar 21. Dot hitam merupakan nilai dari data observasi, garis biru merupakan nilai hasil prediksi, dan garis arsiran biru menggambarkan *uncertainty interval*.



Gambar 21. Visualisasi hasil prediksi

8) Mengembalikan komponen hasil prediksi: Langkah selanjutnya yang perlu dilakukan adalah menampilkan visualisasi komponen hasil prediksi. Hal ini membantu dalam memperlihatkan bagaimana pola harian dari data *time series* berkontribusi dalam membuat prediksi seperti pada gambar 22.

Terlihat dri visualisasi bahwa komponen tren bergerak secara positif, karena seiring waktu garis terus bergerak naik. Pada visualisasi kedua menggambarkan bahwa pola kenaikan pengunjung dimulai dari hari Jumat hingga puncaknya pada hari Minggu, kemudian pola penurunan dimulai setiap hari Senin hingga Kamis. Pola ini selalu berulang setiap minggu



Gambar 22. Visualisasi komponen hasil prediksi

9) Cross-validation: Prophet menyertakan fungsi untuk mengukur tingkat kesalahan / *error* dari prediksi berdasarkan data historis. Hal ini dilakukan dengan menentukan titik *cutoff* dari data historis, dan menyesuaikan model dengan menggunakan data hanya sampai titik batas tersebut. Kemudian dapat membandingkan nilai prediksi dengan nilai sebenarnya. *Cross-validation* dapat dilakukan seperti pada program 22.

```
from fbprophet.diagnostics import
cross_validation
df_cv = cross_validation(my_model, initial='15
days', period='3 days', horizon='5 days')
```

Program 22. *Cross-validation prophet*

Penentuan *cutoff* sebesar 3 hari, data awal (*initial*) sebesar 15 hari (setidaknya ada 2 pola *seasonal* dalam data awal yang digunakan), dan *forecast horizon* sebesar 5 hari (banyaknya hari yang dilakukan prediksi). Ketentuan ini mengikuti *default* dari *prophet*, di mana *initial training periods* memiliki besaran tiga kali *horizon*, dan *cutoff period* memiliki besaran setengah dari *horizon*. Hasil yang diperoleh adalah berupa *dataframe* dengan nilai prediksi dan nilai sebenarnya seperti pada tabel IV.

TABEL IV

DATAFRAME HASIL CROSS VALIDATION

	ds	yhat	yhat_low	yhat_up	y	cutoff
0	2020-10-10	3929.88	3346.55	44567.04	3511	2020-10-09
1	2020-10-11	4872.39	4266.06	5502.89	4292	2020-10-09
2	2020-10-12	2493.93	1855.69	3087.21	2233	2020-10-09
3	2020-10-13	2461.46	1876.59	3068.40	1834	2020-10-09
4	2020-10-14	2567.99	1989.98	3180.33	1965	2020-10-09

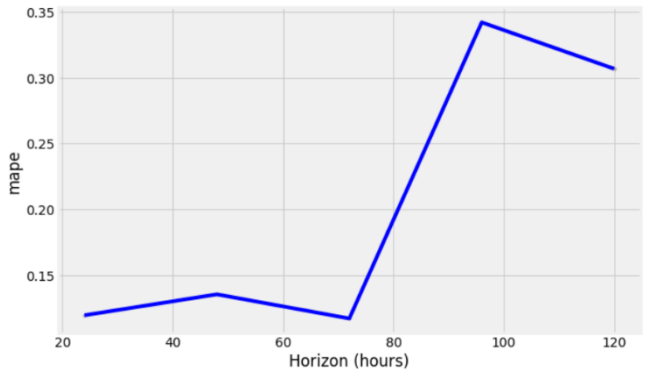
p-ISSN : 2443-2210
e-ISSN : 2443-2229

10) Performance metric: tools ini dapat digunakan untuk menghitung beberapa statistik dari kinerja / performa prediksi (*yhat*, *yhat_lower*, *yhat_upper*, dibandingkan dengan *y*), sebagai fungsi jarak dari batas *cutoff* (seberapa jauh prediksi masa depan). Hal ini dapat dilakukan dengan program 23. Hasil menunjukkan bahwa *horizon* dengan jumlah 3 hari memberikan performa paling baik, *RMSE* 260.93 dengan *MAPE* sebesar 11.68%.

```
from fbprophet.diagnostics import
performance_metrics
df_p = performance_metrics(df_cv)
```

Program 24. *Performance metric prophet*

11) Visualisasi cross-validation: Hasil dari *performance metric* yang diperoleh juga dapat divisualisasikan sehingga memudahkan untuk mengetahui mana yang terbaik seperti pada gambar 23.

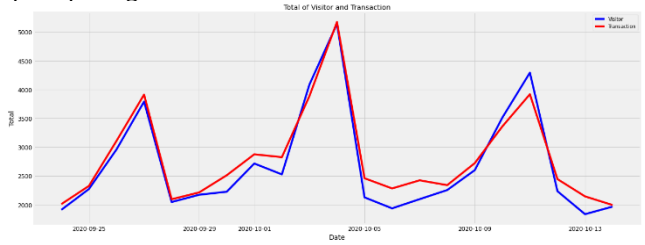


Gambar 23. Visualisasi *performance metric*

12) Data transaksi: Sama halnya dengan data jumlah pengunjung. Data transaksi juga dilakukan analisis menggunakan langkah-langkah yang telah disebutkan di poin-poin sebelumnya. Hasil analisis menunjukkan bahwa performa dari prediksi terbaik dihasilkan oleh *horizon* dengan jumlah 3 hari dengan *RMSE* sebesar 272.61 dan *MAPE* 11.15%.

F. Membandingkan Data Visitor dengan Transaction

Setelah melakukan analisis *time series* terhadap data jumlah pengunjung dan data transaksi, kedua data tersebut memiliki pola kenaikan dan penurunan yang serupa. Akan tetapi, apakah kenaikan dan penurunan tersebut sama persis? Untuk membandingkan pola keduanya, dapat dilakukan visualisasi seperti pada gambar 24.



Gambar 24. Visualisasi kedua data

Untuk menjawab pertanyaan tersebut, dapat dilakukan pengujian statistik seperti uji *t-test*. Namun sebelum melakukan pengujian tersebut, perlu dicek apakah kedua data memenuhi standar pengujian:

- Kedua data memiliki nilai varians yang sama (homogen);
- Kedua data berdistribusi normal

Oleh karena itu, untuk memenuhi standar pertama, dapat dilakukan pengujian statistik Levene seperti pada program 25.

```
# Uji kesamaan varians kedua data
stats.levene(Visitor['Total'],
Transaction['Total'])

# Output
LeveneResult(statistic=0.07626, pvalue=0.78385)
```

Program 25. Uji *Levene*

Nilai p-*value* yang diperoleh dari pengujian *Levene* menunjukkan nilai 0,78 yang lebih besar dari 0,05 sehingga menunjukkan bahwa kedua data memiliki nilai *varians* yang sama (homogen). Selanjutnya adalah melakukan pengujian normalitas untuk mengetahui kedua data berdistribusi normal atau tidak dengan uji statistik *Saphiro* pada program 26.

```
# Uji Normalitas Visitor
stats.saphiro(Visitor['Total'])
# Output
Result(statistic=0.81167, pvalue=0.00099)

# Uji Normalitas Transaction
stats.saphiro(Transaction['Total'])
# Output
Result(statistic=0.83638, pvalue=0.00249)
```

Program 26. Uji normalitas *saphiro*

Nilai p-*value* yang diperoleh dari pengujian normalitas *Saphiro* menunjukkan nilai yang kurang dari 0,05 untuk kedua data. Oleh karena itu, kedua data tidak memenuhi standar normalitas untuk melakukan pengujian *t-test*. Pengujian hipotesis yang dapat dilakukan dengan kondisi seperti ini adalah pengujian statistic non-parametrik, seperti pengujian *Wilcoxon* pada program 27.

```
# Uji Wilcoxon

from scipy.stats import wilcoxon
stat, p = wilcoxon(Visitor['Total'],
Transaction['Total'])
print('statistics=%.3f, p=%.3f'%(stat, p))

# Output
statistic=45.000, p=0.014
```

Program 27. Uji *Wilcoxon*

Dari pengujian *Wilcoxon*, diperoleh nilai *p-value* sebesar 0,014 yang lebih rendah dari 0,05. Hal ini berarti bahwa pola data jumlah pengunjung tidak sama dengan pola data transaksi (H0 ditolak).

JuTISI
Jurnal Teknik Informatika dan Sistem Informasi

V. SIMPULAN

Melalui eksperimen pada penelitian ini, ada beberapa hal yang dapat dicapai. Pertama, dapat diperoleh gambaran mengenai jumlah pengunjung toko *retail X* melalui pemanfaatan terknologi pendeteksian citra *computer vision*. Kedua, diperoleh gambaran kondisi performa toko secara umum melalui *retail conversion rate*. Ketiga, visualisasi data jumlah pengunjung dan data transaksi juga menggambarkan kondisi-kondisi pada waktu kapan toko *retail X* ramai dipenuhi oleh pengunjung, ini dapat dimanfaatkan oleh toko *retail X* untuk mengatur penempatan jumlah *staff* pada setiap harinya.

Keempat, prediksi mengenai proyeksi jumlah pengunjung dan potensi terjadinya transaksi dapat dilakukan. Hasil prediksi baik untuk data jumlah pengunjung maupun data transaksi dapat divisualisasikan dan memiliki hasil yang mendekati data observasi, walaupun semakin jauh prediksi dilakukan, maka akan semakin berkurang interval kepercayaan nya. Hal ini dikarenakan rentang waktu data observasi yang terbilang pendek.

Kelima, analisis *time series* terhadap data jumlah pengunjung dan data transaksi menunjukkan hasil performa yang lebih baik dalam pembentukan model prediksi menggunakan metode *exponential smoothing Holt-Winter* dan kriteris penilaian *AICc* jika dibandingkan dengan metode *SARIMAX* dan kriteria penilaian *AIC*. Metode *Holt-Winter* memberikan optimalisasi paramater terhadap faktor level, tren, dan *seasonal* dengan memberikan pembobotan dari setiap model yang dibentuk.

Selain itu, kriteria penilaian *AICc* mengurangi bias yang ditimbulkan oleh data untuk melakukan prediksi. Kriteria penghitungan ini lebih mendalam jika dibandingkan dengan kriteria *AIC*, sehingga dapat disebut bahwa *AICc* memberikan penghitungan lebih mendetail (*second order estimate*). Pemilihan kriteria ini lebih pas bagi ukuran sampel data yang kecil ($n < 30$).

Analisis *time series* dengan metode *FB Prophet* juga memberikan hasil yang lebih baik jika dibandingkan dengan metode *SARIMAX*. Namun hal yang menarik adalah untuk data jumlah pengujung, metode *FB Prophet* memberikan hasil performa prediksi lebih buruk jika dibandingkan dengan metode *Holt-Winter* (*RMSE Prophet Visitor* 260.93 dengan *RMSE Holt-Winter Visitor* 178.1). Lain halnya dengan data transaksi yang memberikan hasil performa prediksi lebih baik menggunakan metode *FB Prophet* jika dibandingkan dengan metode *Holt-Winter* (*RMSE Prophet Transaction* 272.61 dengan *RMSE Holt-Winter Transaction* 292.6).

Terakhir, penelitian ini menjawab bahwa pola antara kedua data tidak sama, walaupun secara sekilas memiliki pola kenaikan dan penurunan yang serupa. Hal ini diketahui setelah membandingkan kedua pola data, dan mengujinya dengan metode *Wilcoxon* yang memberikan hasil *p-value* senilai 0.014.

DAFTAR PUSTAKA

[1] CountBOX. (2017) Footfall traffic: why it matters. [Online]. Tersedia: http://countbox.us/footfall-traffic-matters/

[2] O. Perdikaki, S. Kesavan, &J. M. Swaminathan, "Effect of Traffic on Sales and Conversion Rates of Retail Stores," *informs Manufacturing & Service Operations Management*, vol. 14, issue 1, pp. 145-162, 2012.

[3] C. Y. Yiu &H. C. Ng, "Buyers-to-shoppers ratio of shopping malls: A probit study in Hong Kong," *Journal of Retailing and Consumer Services*, vol. 17, issue 5, pp. 349-354, 2010.

[4] Localistico. (2019). Footfall analysis: counting customers in store to increase engagement. [Online]. Tersedia: http://localistico.com/blog/footfall-analysis-local-marketing/

[5] Ipsos Retail Performance. (2017). How to Calculate Retail Conversion Rate. [Online]. Tersedia: http://ipsosreetailperformance.com/en/insights/calculate-stores-conversion-rate/

[6] Towards data science. (2019). Computer Vision. Here's A Look Why It's So Awesome. [Online]. Tersedia: towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

[7] D. Forsyth & J. Ponce, *Computer Vision: A Modern Approach*, 2nd ed., Upper Saddle River, New Jersey: Pearson Education, Inc., 2011.

[8] J. E. Solem, *Programming Computer Vision with Python: Tools and algorithms for analyzing images*, 1st ed., O'Reilly Media, Inc., 2012.

[9] GeeksforGeeks. (2019). OpenCV – Overview. [Online]. Tersedia: geeksforgeeks.org/opencv-overview/

[10] OpenCV. (2020). The OpenCV Website. [Online]. Tersedia: opencv.org/about/

[11] D. M. C. Machado, "People Counting System Using Existing Surveillance Video Camera," *ISEP DM Electrical and Computer Engineering.*, 2011.

[12] K. Terada, D. Yoshida, S. Oe & J. Yamaguchi, "A Method of Counting the Passing People by Using the Stereo Image," *International Conference on Image Processing (Cat. 99CH36348)*, vol. 2, pp. 338-342, 1999.

[13] D. Beymer, "Person Counting Using Stereo," *Workshop on Human Motion*, pp. 127-133, 2000.

[14] Pyimagesearch. (2017). Object Detection with Deep Learning and OpenCV. [Online]. Tersedia: pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/

[15] Optimizely. (2020). Conversion Rate. [Online]. Tersedia: optimizely.com/optimization-glossary/conversion-rate/

[16] Axper. (2016). The Power of The Performance on Traffic. [Online]. Tersedia: axper.com/the-power-of-the-performance-on-traffic/

[17] Erply. (2020). How to Measure Retail Performance?. [Online]. Tersedia: erply.com/how-to-measure-retail-performance-5-essential-metrics/

[18] BetterProgramming. (2019). Fundamentals of Time Series Data and Forecasting Trend, seasonality, and cycles in a few minutes. [Online]. Tersedia: medium.com/better-programming/fundamentals-of-time-series-data-and-forecasting-15e9490b2

[19] Towards data science. (2019). Time series-Introduction. [Online]. Tersedia: towardsdatascience.com/time-series-information-7484bc25739a

[20] R. J. Hyndman & G. Athanasopoulos, Forecasting: Principles and Practice, 2nd ed, Monash University, Australia: Otexts, 2018.

[21] Pyimagesearch. (2018). OpenCV People Counter. [Online]. Tersedia: pyimagesearch.com/2018/08/13/opencv-people-counter/

[22] Caffe. (2017). Blobs, Layers, and Nets: anatomy of a Caffe model. [Online]. Tersedia: caffe.berkeleyvision.org/tutorial/net_layer_blob.html/

[23] Pyimagesearch. (2018). Simple Object Tracking with OpenCV [Online]. Tersedia: pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/ [Diakses 4 Maret 2020].

[24] P.J. Brockwell dan R.A. Davis, Introduction to Time Series and Forecasting, 2nd Edition., Department of Statistics Colorado State University, New York: Springer, 2002.

[25] Github, "Time Series Forecasting in Python & R, Part 2 (Forecasting)," 21 April 2020. [Online]. Available: pawarbi.github.io/blog/forecasting/r/python/rpy2/altair/fbprophet/s