
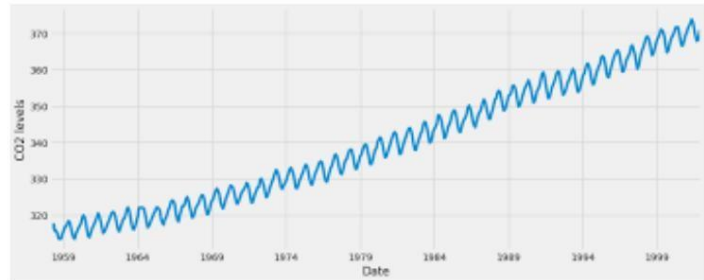
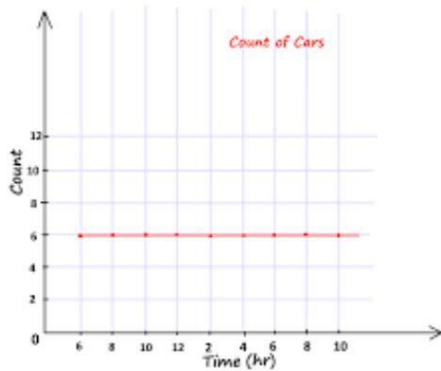


Analisis Vidhya

 kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6132962-pengenalan-seri-waktu

Pengantar Rangkaian Waktu

Menurut Anda, manakah dari berikut ini yang merupakan contoh deret waktu? Kalaupun belum tahu, coba tebak.



Rangkaian Waktu umumnya adalah data yang dikumpulkan dari waktu ke waktu dan bergantung padanya.

Di sini kita melihat bahwa jumlah mobil tidak bergantung pada waktu, oleh karena itu ini bukan deret waktu.

Meskipun tingkat CO2 meningkat terhadap waktu, maka ini adalah deret waktu.

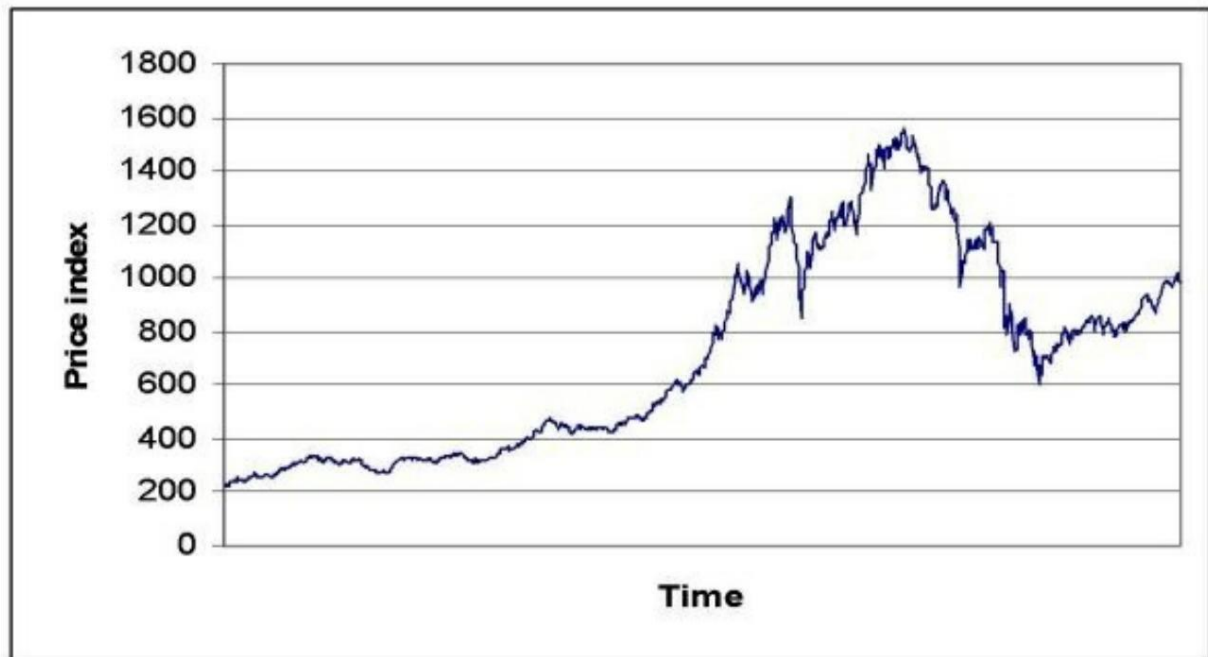
Sekarang mari kita lihat definisi formal Time Series.

Serangkaian titik data yang dikumpulkan dalam urutan waktu dikenal sebagai deret waktu. Sebagian besar rumah bisnis menggunakan data deret waktu untuk menganalisis jumlah penjualan tahun depan, lalu lintas situs web, jumlah lalu lintas, jumlah panggilan yang diterima, dll. Data deret waktu dapat digunakan untuk peramalan.

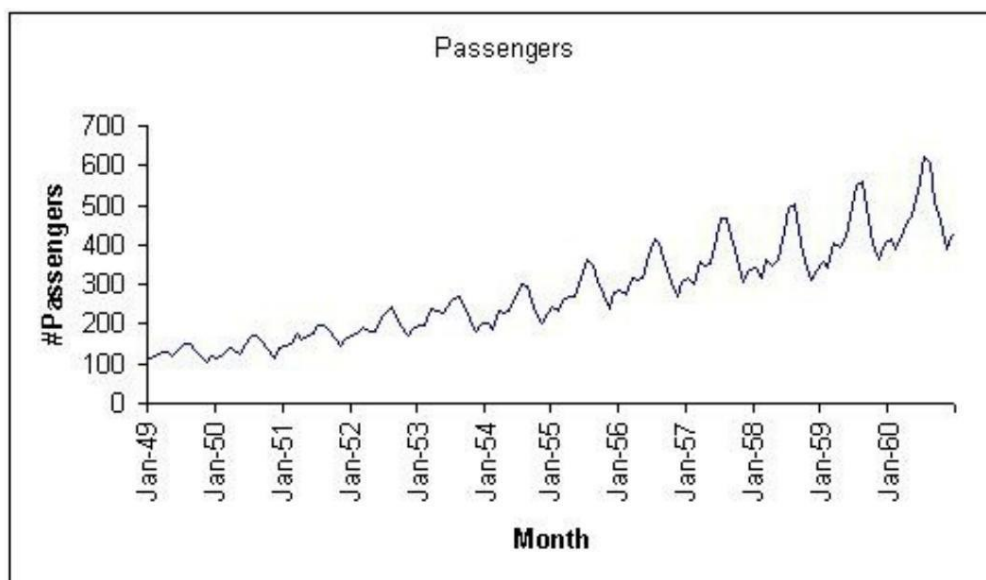
Tidak semua data yang dikumpulkan sehubungan dengan waktu mewakili rangkaian waktu.

Beberapa contoh deret waktu adalah:

Harga Saham :

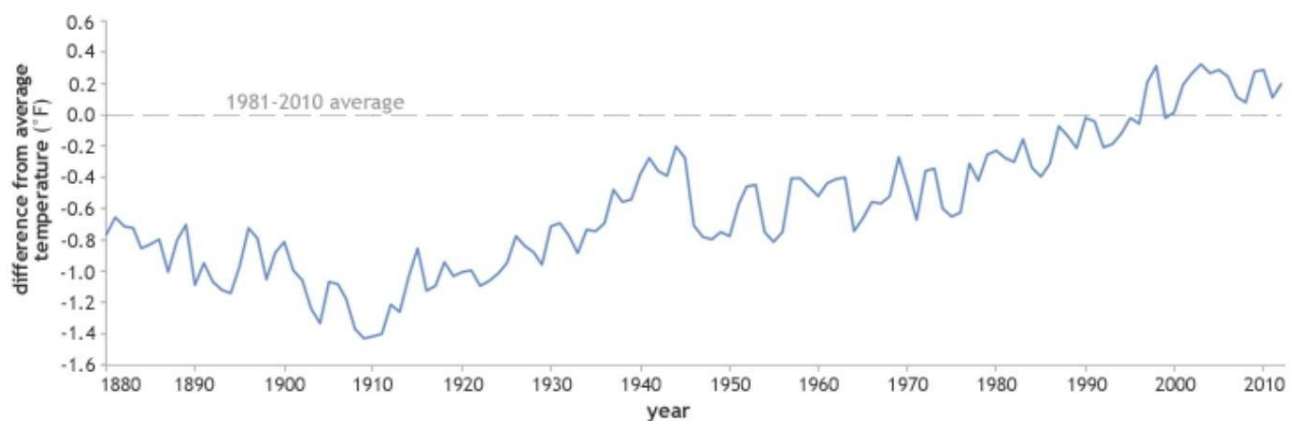


Jumlah Penumpang suatu maskapai penerbangan :



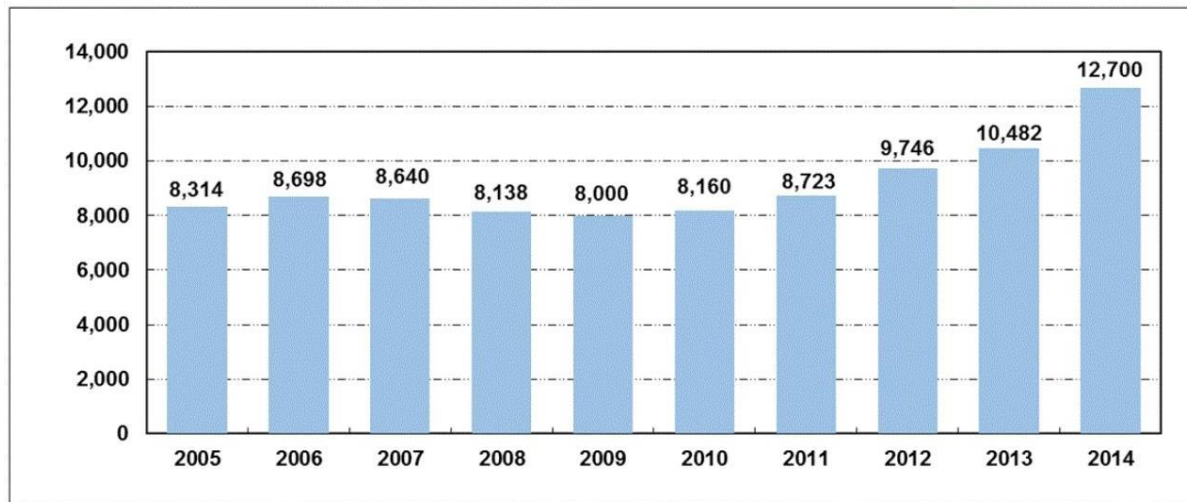
Suhu dari waktu ke waktu:

Yearly sea-surface temperature anomaly 1880-2012



Jumlah pengunjung di sebuah hotel :

Visitors to Universal Studios Japan (Unit: million)



Source: Universal Studios Japan

Sekarang kita sudah bisa membedakan antara data Time Series dan non Time Series, mari kita jelajahi Time Series lebih jauh.

Analisis Vidhya

kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133008-komponen-of-a-time-series

Sekarang setelah kita memahami apa itu deret waktu dan perbedaan antara deret waktu dan non deret waktu, sekarang mari kita lihat komponen deret waktu.

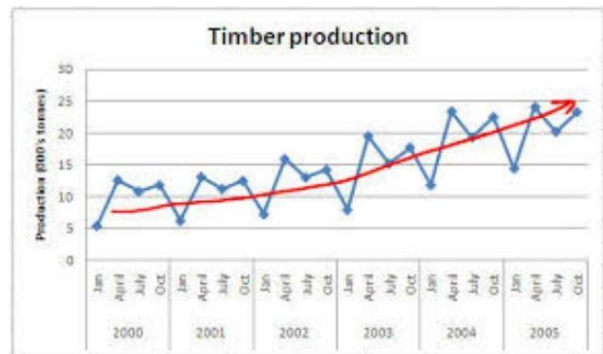
Komponen Rangkaian Waktu

1. **Tren** : Tren adalah arah umum berkembang atau berubahnya sesuatu.

Jadi kami melihat tren peningkatan dalam rangkaian waktu ini. Kita dapat melihat bahwa jumlah penumpang terus meningkat seiring dengan berjalannya waktu. Mari kita visualisasikan tren deret waktu:

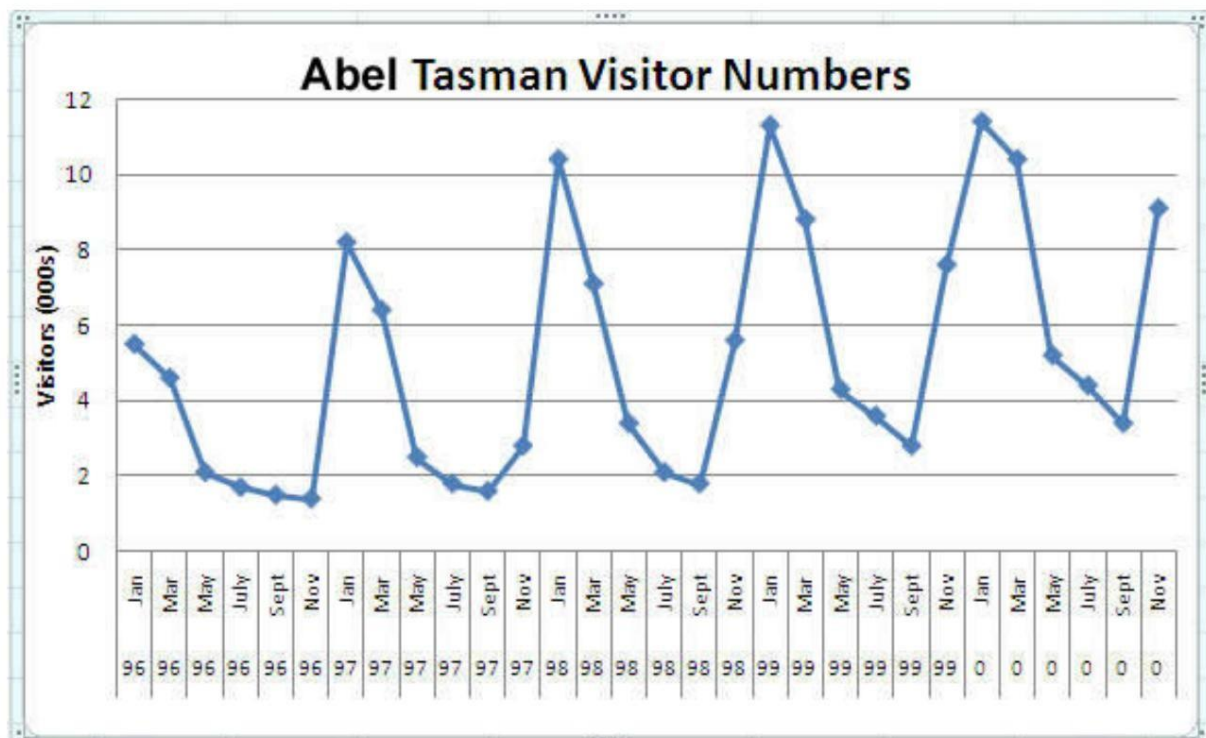
Contoh

Di sini garis merah mewakili tren peningkatan deret waktu.



2. **Musiman** : Pola lain yang jelas juga dapat dilihat pada waktu di atas

deret, yaitu pola yang berulang pada selang waktu teratur yang disebut musiman. Setiap perubahan atau pola yang dapat diprediksi dalam suatu rangkaian waktu yang berulang atau berulang dalam jangka waktu tertentu dapat dikatakan sebagai musiman. Mari kita visualisasikan musiman deret waktu:



Contoh

Kita dapat melihat bahwa deret waktu mengulangi polanya setiap 12 bulan, yaitu terdapat puncak setiap tahun pada bulan Januari dan titik terendah setiap tahun pada bulan September, sehingga deret waktu ini mempunyai musiman selama 12 bulan.

Perbedaan antara masalah deret waktu dan regresi

Di sini Anda mungkin berpikir bahwa karena variabel target bersifat numerik, maka variabel tersebut dapat diprediksi menggunakan teknik regresi, namun masalah deret waktu berbeda dari masalah regresi dalam hal berikut:

- Perbedaan utamanya adalah deret waktu bergantung pada waktu. Jadi asumsi dasar model regresi linier bahwa observasi bersifat independen tidak berlaku dalam kasus ini.
- Seiring dengan tren yang meningkat atau menurun, sebagian besar Time Series memiliki beberapa bentuk tren musiman, yaitu variasi yang spesifik pada jangka waktu tertentu.

Jadi, memprediksi suatu deret waktu dengan menggunakan teknik regresi bukanlah pendekatan yang baik.

Analisis deret waktu terdiri dari metode untuk menganalisis data deret waktu untuk mengekstrak statistik yang bermakna dan karakteristik data lainnya. Peramalan deret waktu adalah penggunaan model untuk memprediksi nilai masa depan berdasarkan nilai yang diamati sebelumnya.

Analisis Vidhya

kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133008-komponen-of-a-time-series

Sekarang setelah kita memahami apa itu deret waktu dan perbedaan antara deret waktu dan non deret waktu, sekarang mari kita lihat komponen deret waktu.

Komponen Rangkaian Waktu

1. **Tren** : Tren adalah arah umum berkembang atau berubahnya sesuatu.

Jadi kami melihat tren peningkatan dalam rangkaian waktu ini. Kita dapat melihat bahwa jumlah penumpang terus meningkat seiring dengan berjalannya waktu. Mari kita visualisasikan tren deret waktu:

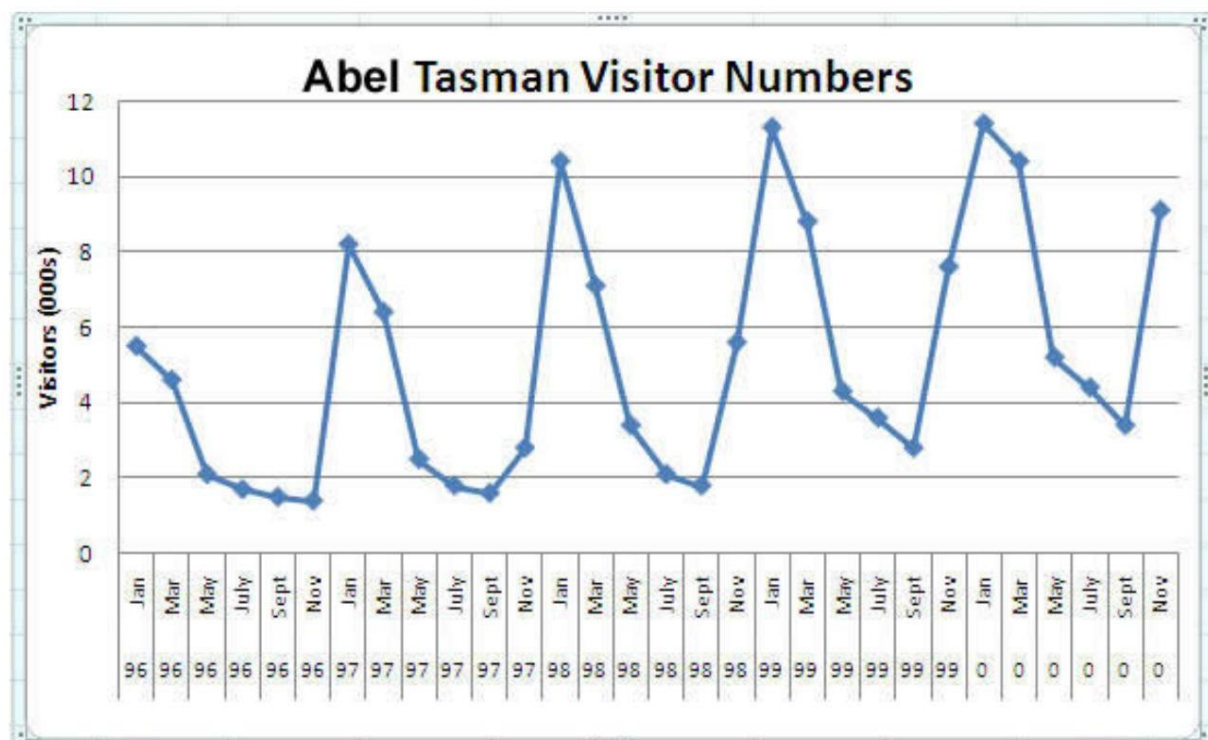
Contoh

Di sini garis merah mewakili tren peningkatan deret waktu.



2. **Musiman** : Pola lain yang jelas juga dapat dilihat pada waktu di atas

deret, yaitu pola yang berulang pada selang waktu tertentu yang disebut musiman. Setiap perubahan atau pola yang dapat diprediksi dalam suatu rangkaian waktu yang berulang atau berulang dalam jangka waktu tertentu dapat dikatakan sebagai musiman. Mari kita visualisasikan musiman deret waktu:



Contoh

Kita dapat melihat bahwa deret waktu mengulangi polanya setiap 12 bulan, yaitu terdapat puncak setiap tahun pada bulan Januari dan titik terendah setiap tahun pada bulan September, sehingga deret waktu ini mempunyai musiman selama 12 bulan.

Perbedaan antara masalah deret waktu dan regresi


Di sini Anda mungkin berpikir bahwa karena variabel target bersifat numerik, maka variabel tersebut dapat diprediksi menggunakan teknik regresi, namun masalah deret waktu berbeda dari masalah regresi dalam hal berikut:

- Perbedaan utamanya adalah deret waktu bergantung pada waktu. Jadi asumsi dasar model regresi linier bahwa observasi bersifat independen tidak berlaku dalam kasus ini.
- Seiring dengan tren yang meningkat atau menurun, sebagian besar Time Series memiliki beberapa bentuk tren musiman, yaitu variasi yang spesifik pada jangka waktu tertentu.

Jadi, memprediksi suatu deret waktu dengan menggunakan teknik regresi bukanlah pendekatan yang baik.

Analisis deret waktu terdiri dari metode untuk menganalisis data deret waktu untuk mengekstrak statistik yang bermakna dan karakteristik data lainnya. Peramalan deret waktu adalah penggunaan model untuk memprediksi nilai masa depan berdasarkan nilai yang diamati sebelumnya.

Analisis Vidhya

 [kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133009-pernyataan masalah](https://kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133009-pernyataan-masalah)

Sekarang setelah kita memahami berbagai komponen deret waktu, mari kita lihat rumusan masalah yang akan kita selesaikan dalam kursus ini.

Pernyataan Masalah

Unicorn Investors ingin berinvestasi pada bentuk transportasi baru - JetRail.

JetRail menggunakan teknologi propulsi Jet untuk menjalankan rel dan memindahkan orang dengan kecepatan tinggi! Investasi tersebut hanya akan masuk akal jika mereka dapat memperoleh lebih dari 1 Juta pengguna bulanan dalam 18 bulan ke depan. Untuk membantu Unicorn Ventures dalam mengambil keputusan, Anda perlu memperkirakan lalu lintas di JetRail untuk 7 bulan ke depan. Anda diberikan data lalu lintas JetRail sejak awal dalam file pengujian.

Anda bisa mendapatkan datasetnya [di sini https://datahack.analyticsvidhya.com/contest/practice-problem-time-series-2/](https://datahack.analyticsvidhya.com/contest/practice-problem-time-series-2/)

Disarankan untuk melihat kumpulan data setelah menyelesaikan bagian pembuatan hipotesis.

Analisis Vidhya

 course.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133010- daftar isi

Sekarang kita sudah memiliki kumpulan data serta pemahaman tentang rumusan masalah, mari kita lihat langkah-langkah yang akan kita ikuti dalam kursus ini untuk menyelesaikan masalah yang ada.

Daftar isi


a) Memahami Data:

- 1) Pembuatan Hipotesis
 - 2) Mempersiapkan sistem dan memuat data
 - 3) Struktur dan Konten Kumpulan Data
 - 4) Ekstraksi Fitur
 - 5) Analisis Eksplorasi
-

b) Peramalan menggunakan Teknik Pemodelan Berganda:

- 1) Membagi data menjadi bagian pelatihan dan validasi
 - 2) Teknik pemodelan
 - 3) Model Tren Linier Holt pada deret waktu harian
 - 4) Model Holt Winter pada rangkaian waktu harian
 - 5) Pengenalan model ARIMA
 - 6) Penyetelan parameter untuk model ARIMA
 - 7) Model SARIMAX pada deret waktu harian
-

Analisis Vidhya

 [kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133013-generasi hipotesis](https://kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133013-generasi-hipotesis)

Kita akan mulai dengan langkah pertama, yaitu Pembuatan Hipotesis. Pembuatan Hipotesis adalah proses membuat daftar semua faktor yang mungkin mempengaruhi hasil.

Pembuatan hipotesis dilakukan sebelum melihat data untuk menghindari bias yang mungkin timbul setelah observasi.

1) Pembuatan Hipotesis

Pembuatan hipotesis membantu kita menunjukkan faktor-faktor yang mungkin mempengaruhi variabel dependen kita. Di bawah ini adalah beberapa hipotesis yang menurut saya dapat mempengaruhi jumlah penumpang (variabel terikat untuk masalah deret waktu ini) di JetRail:

1. Akan terjadi peningkatan lalu lintas seiring berjalannya waktu.

Penjelasan - Populasi secara umum memiliki tren peningkatan seiring berjalannya waktu, jadi saya perkirakan lebih banyak orang akan bepergian dengan JetRail. Selain itu, umumnya perusahaan memperluas bisnis mereka dari waktu ke waktu sehingga menghasilkan lebih banyak pelanggan yang bepergian melalui JetRail.

2. Lalu lintas akan tinggi dari bulan Mei sampai Oktober.

Penjelasan – Kunjungan wisatawan umumnya meningkat pada masa liburan ini.

3. Lalu lintas pada hari kerja akan lebih banyak dibandingkan pada akhir pekan/hari libur.


Penjelasan - Orang-orang akan pergi ke kantor pada hari kerja dan karenanya kemacetan akan terjadi lagi

4. Lalu lintas pada jam sibuk akan tinggi.

Penjelasan - Orang akan bepergian untuk bekerja, kuliah.

Kami akan mencoba memvalidasi setiap hipotesis ini berdasarkan kumpulan data. Sekarang mari kita lihat kumpulan datanya.

Analisis Vidhya

 kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133032-menyiapkan-sistem-dan-memuat-data

Setelah membuat hipotesis, kami akan mencoba memvalidasinya. Sebelumnya kami akan mengimpor semua paket yang diperlukan.

2) Mempersiapkan sistem dan memuat data

Versi:

- Python = 3.7
- Panda = 0.20.3
- belajar sklearn = 0.19.1

Sekarang kita akan mengimpor semua paket yang akan digunakan di seluruh notebook.

```
import pandas as pd
import numpy as np # Untuk perhitungan matematis
import matplotlib.pyplot as plt # Untuk memplot grafik dari datetime
import datetime # Untuk mengakses datetime dari pandas
import Series # Untuk mengerjakan peringatan
import inline
import matplotlib

# Untuk mengabaikan peringatan
warnings.filterwarnings("ignore")
```


Sekarang mari kita membaca data latih dan uji

```
kereta=pd.read_csv("Train_SU63lSt.csv")
tes=pd.read_csv("Test_0qrQsBZ.csv")
```

Mari kita buat salinan data latih dan uji sehingga meskipun kita melakukan perubahan pada dataset tersebut, kita tidak kehilangan dataset aslinya.

```
train_original=kereta.copy()
test_original=test.copy()
```

Analisis Vidhya

 kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133080-struktur-dan-konten-kumpulan-data

Setelah memuat data, mari kita lihat sekilas kumpulan data untuk mengetahui data kita lebih baik.

3) Struktur dan Konten Kumpulan Data

Mari selami lebih dalam dan lihat kumpulan datanya. Pertama-tama mari kita lihat fitur-fitur dalam kumpulan data pelatihan dan pengujian.

kereta.kolom, tes.kolom

```
(Indeks(['ID', 'Datetime', 'Count'], dtype='object'),  
 Indeks(['ID', 'Datetime'], dtype='objek'))
```

Kami memiliki ID, Tanggal Waktu dan jumlah penumpang yang sesuai di file kereta. Untuk file pengujian kami hanya memiliki ID dan Tanggal waktu sehingga kami harus memprediksi Hitungan untuk file pengujian.

Mari kita pahami setiap fiturnya terlebih dahulu:

- ID adalah nomor unik yang diberikan pada setiap titik pengamatan.
- Datetime adalah waktu setiap observasi.
- Hitungan adalah jumlah penumpang yang sesuai dengan setiap Datetime.

Mari kita lihat tipe data setiap fitur.

kereta.dtypes, tes.dtypes

```
(ID int64  
 Objek tanggal dan waktu  
 Hitung int64  
dtype: objek, ID int64  
Tipe objek tanggalwaktu:  
objek)
```

- ID dan Count dalam format integer sedangkan Datetime dalam format objek untuk file kereta.
- Id dalam bilangan bulat dan Datetime dalam format objek untuk file pengujian.


Sekarang kita akan melihat bentuk datasetnya.

kereta.bentuk, tes.bentuk

```
((18288, 3), (5112, 2))
```

Kami memiliki 18288 catatan berbeda untuk Jumlah penumpang di set kereta dan 5112 dalam pengujian

Analisis Vidhya

 [kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133138-ekstraksi fitur](https://kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133138-ekstraksi%20fitur)

Sekarang kami akan mengekstrak lebih banyak fitur untuk memvalidasi hipotesis kami.

4) Ekstraksi Fitur

Kami akan mengekstrak waktu dan tanggal dari Datetime. Kita telah melihat sebelumnya bahwa tipe data Datetime adalah objek. Jadi pertama-tama kita harus mengubah tipe data ke format datetime jika tidak, kita tidak dapat mengekstrak fitur darinya.

```
kereta['Datetime'] = pd.to_datetime(train.Datetime,format='%d-%m-%Y %H:%M') test['Datetime'] =  
pd.to_datetime(test.Datetime,format= '%d-%m-%Y %H:%M') test_original['Datetime'] =  
pd.to_datetime(test_original.Datetime,format='%d-%m-%Y %H:%M') train_original ['Datetime'] =  
pd.to_datetime(train_original.Datetime,format='%d-%m-%Y %H:%M')
```

Kami membuat beberapa hipotesis tentang pengaruh jam, hari, bulan dan tahun terhadap jumlah penumpang. Jadi, mari kita ekstrak tahun, bulan, hari, dan jam dari Datetime untuk memvalidasi hipotesis kita.

untuk saya di (kereta, tes, tes_asli, kereta_asli):

```
i['tahun']=i.Tanggalwaktu.dt.tahun  
i['bulan']=i.Tanggalwaktu.dt.bulan  
i['hari']=i. Tanggalwaktu.dt.hari  
i['Jam']=i.Tanggalwaktu.dt.jam
```

Kami juga membuat hipotesis untuk pola lalu lintas pada hari kerja dan akhir pekan. Jadi, mari kita buat variabel akhir pekan untuk memvisualisasikan dampak akhir pekan terhadap lalu lintas.

- Pertama-tama kita akan mengekstrak hari dalam seminggu dari Datetime dan kemudian berdasarkan nilai yang akan kita tetapkan apakah hari itu adalah akhir pekan atau bukan.
- Nilai 5 dan 6 menyatakan bahwa hari tersebut adalah akhir pekan.

```
kereta['hari dalam seminggu']=kereta.api['Datetime'].dt.hari  
minggu temp = kereta['Datetime']
```

Mari kita tetapkan 1 jika hari dalam seminggu adalah akhir pekan dan 0 jika hari dalam seminggu bukan akhir pekan.

def pemohon (baris):

```
    jika baris.hari minggu == 5 atau baris.hari minggu == 6:  
        kembali 1
```

kalau tidak:

```
    kembali 0
```

```
temp2 = kereta['Datetime'].apply(pemohon)  
kereta['akhir pekan']=temp2
```

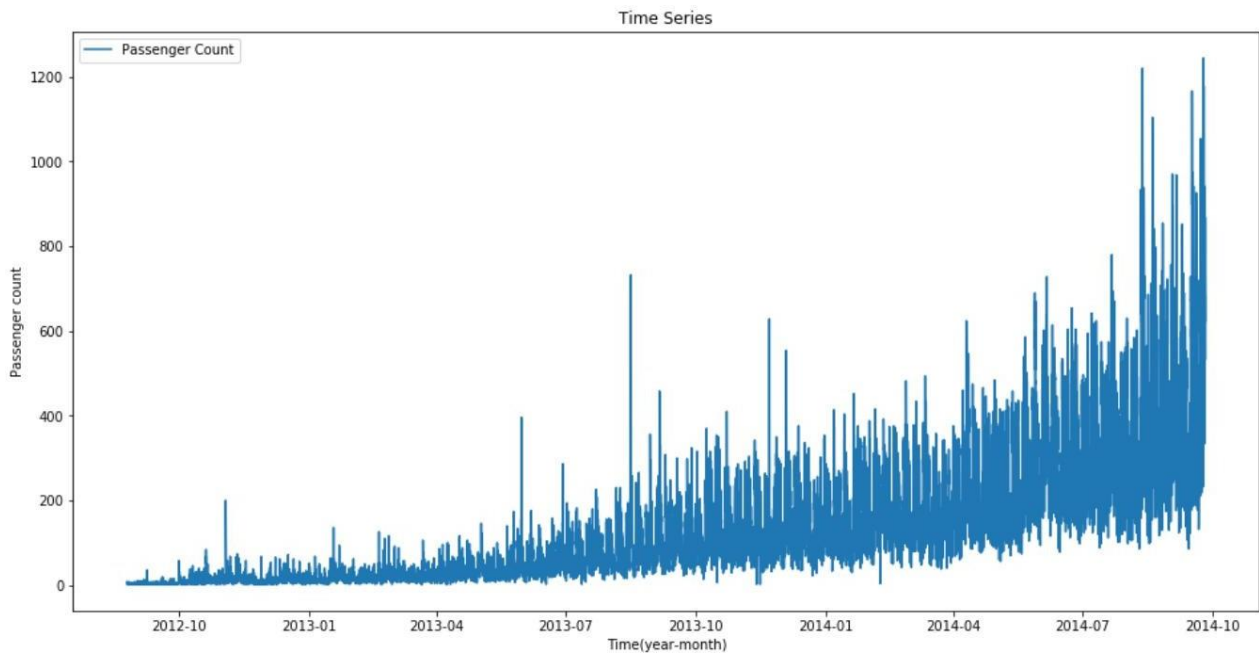
Mari kita lihat rangkaian waktunya.

```

train.index = train['Datetime'] # mengindeks Datetime untuk mendapatkan periode waktu pada sumbu x. df=train.drop('ID',1)
ts = df['Hitungan']             # jatuhkan variabel ID untuk mendapatkan Datetime saja pada sumbu x.
plt.figure(figsize=(16,8))
plt.plot(ts, label='Jumlah
Penumpang') plt.title(' Time Series')
plt.xlabel("Waktu(tahun-
bulan)") plt.ylabel("Jumlah penumpang")
plt.legend(loc='terbaik')

```

<matplotlib.legend.Legend at 0x7f2ad5ce62b0>



Di sini kita dapat menyimpulkan bahwa ada tren peningkatan dalam rangkaian tersebut, yaitu jumlah hitungan yang meningkat terhadap waktu. Kita juga dapat melihat bahwa pada titik-titik tertentu terjadi peningkatan jumlah hitungan secara tiba-tiba. Kemungkinan alasan di balik hal ini adalah pada hari tertentu, karena suatu peristiwa, lalu lintas sedang tinggi.

Kami akan mengerjakan file kereta untuk semua analisis dan akan menggunakan file pengujian untuk peramalan.

Analisis Vidhya

kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6133152-analisis-eksplorasi

Mari kita ingat kembali hipotesis yang kita buat sebelumnya:

- Lalu lintas akan meningkat seiring berjalannya waktu
- Lalu lintas akan tinggi dari Mei hingga Oktober
- Lalu lintas pada hari kerja akan lebih banyak
- Lalu lintas pada jam sibuk akan tinggi

Setelah melihat kumpulan data, sekarang kami akan mencoba memvalidasi hipotesis kami dan membuat kesimpulan lain dari kumpulan data tersebut.

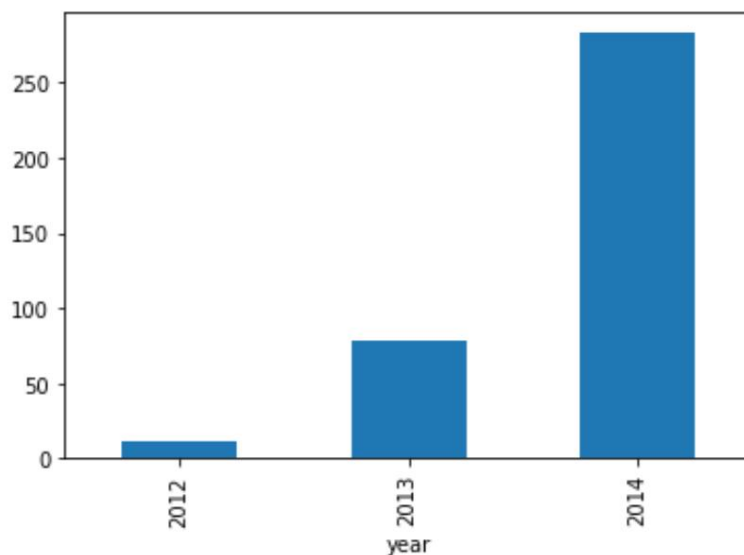
5) Analisis Eksplorasi

Mari kita coba memverifikasi hipotesis kita menggunakan data sebenarnya.

Hipotesis pertama kami adalah lalu lintas akan meningkat seiring berjalannya waktu. Jadi mari kita lihat jumlah penumpang tahunan.

```
train.groupby('tahun')['Hitungan'].mean().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot di 0x7f2cfb0424a8>

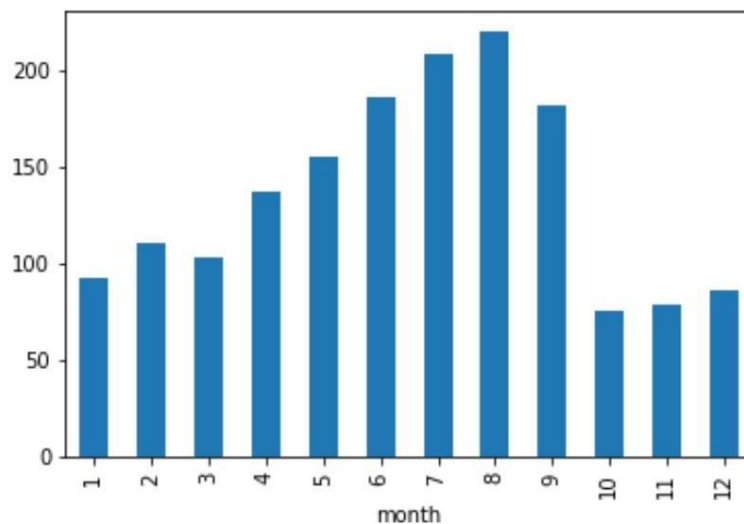


Kami melihat pertumbuhan lalu lintas yang eksponensial terhadap tahun yang memvalidasi hipotesis kami.

Hipotesis kedua kami adalah tentang peningkatan lalu lintas dari bulan Mei hingga Oktober. Jadi, mari kita lihat hubungan antara hitungan dan bulan.

```
train.groupby('bulan')['Hitungan'].mean().plot.bar()
```

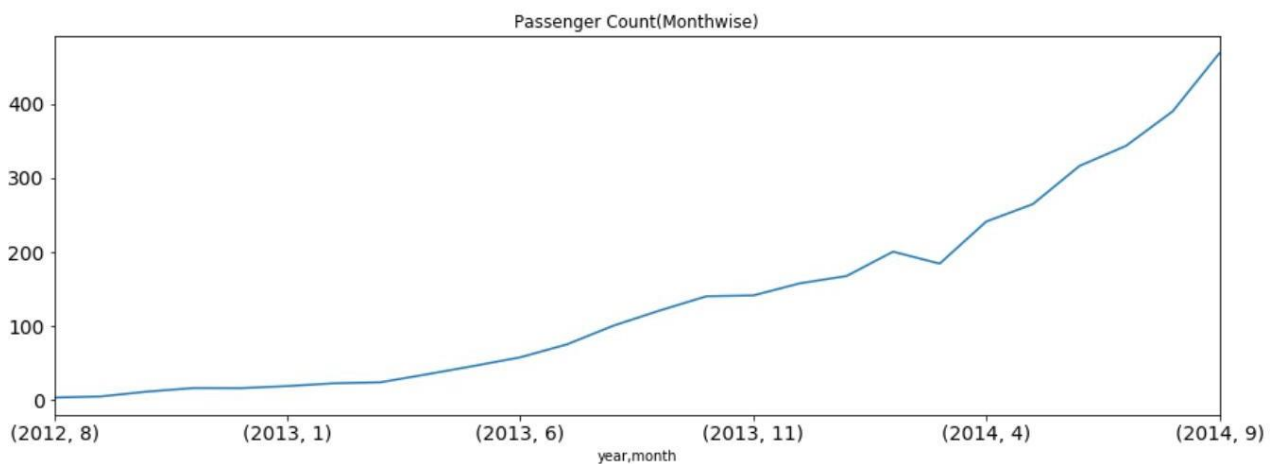

<matplotlib.axes._subplots.AxesSubplot di 0x7f2cfb82a7f0>



Di sini kita melihat penurunan rata-rata jumlah penumpang dalam tiga bulan terakhir. Ini sepertinya tidak benar. Mari kita lihat rata-rata bulanan setiap tahun secara terpisah.

```
temp=train.groupby(['tahun', 'bulan'])['Hitungan'].mean()
temp.plot(figsize=(15,5), title= 'Jumlah Penumpang(Bulanan)', ukuran font=14 )
```

<matplotlib.axes._subplots.AxesSubplot di 0x7f2cfb878eb8>

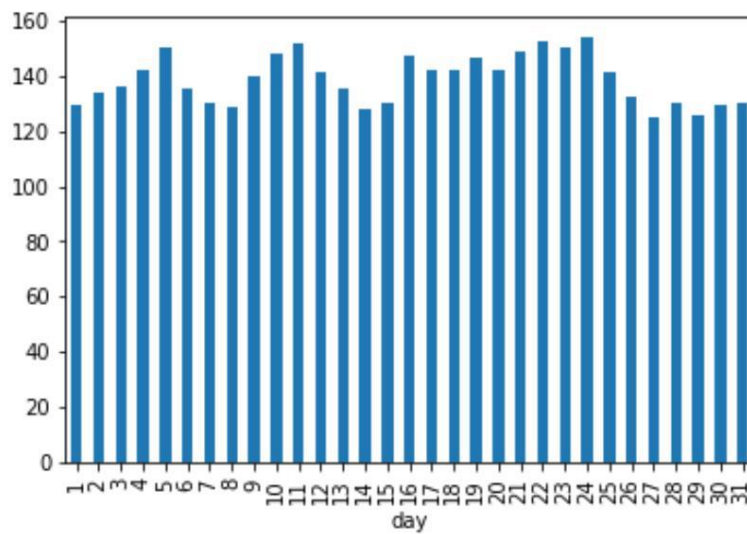


- Kita melihat bahwa bulan 10, 11 dan 12 tidak ada pada tahun 2014 dan nilai rata-rata bulan-bulan tersebut pada tahun 2012 sangat kecil.
- Karena ada tren peningkatan dalam rangkaian waktu kami, nilai rata-rata untuk bulan-bulan lainnya akan lebih besar karena jumlah penumpang yang lebih besar pada tahun 2014 dan kami akan mendapatkan nilai yang lebih kecil untuk 3 bulan ini.
- Pada plot garis di atas kita dapat melihat tren peningkatan jumlah penumpang bulanan dan pertumbuhannya kira-kira eksponensial.

Mari kita lihat rata-rata jumlah penumpang harian.

```
train.groupby('hari')['Hitungan'].mean().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot di 0x7f2cfb74dd68>

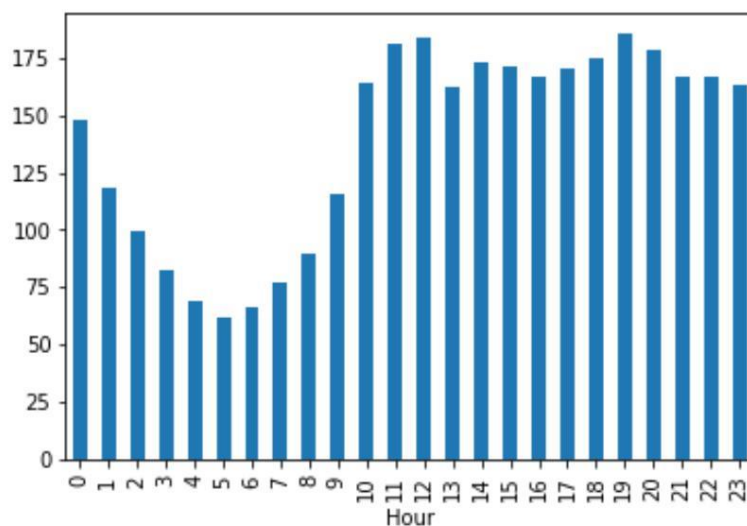


Kami tidak mendapatkan banyak wawasan dari penghitungan penumpang setiap harinya.

Kami juga membuat hipotesis bahwa lalu lintas akan lebih banyak pada jam sibuk. Jadi mari kita lihat rata-rata jumlah penumpang per jam.

```
train.groupby('Jam')['Hitungan'].mean().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot di 0x7f2cfb59d5c0>



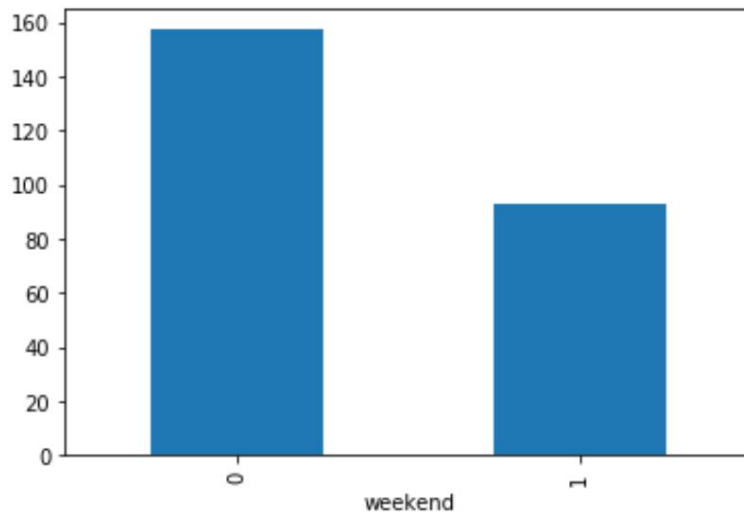
- Dapat disimpulkan bahwa puncak lalu lintas terjadi pada jam 7 malam dan kemudian kita melihat tren penurunan hingga jam 5 pagi.
- Setelah itu jumlah penumpang mulai meningkat lagi dan mencapai puncaknya lagi di antara keduanya

11 pagi dan 12 siang.

Mari kita coba memvalidasi hipotesis kita yang berasumsi bahwa lalu lintas akan lebih banyak pada hari kerja.

```
train.groupby('akhir pekan')['Hitungan'].mean().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot di 0x7f2cfb4d7ac8>



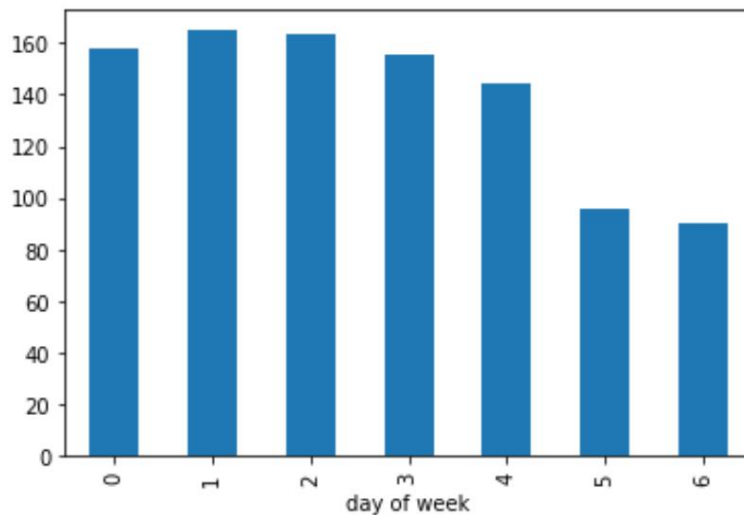
Dapat disimpulkan dari plot di atas bahwa lalu lintas lebih banyak pada hari kerja dibandingkan pada akhir pekan yang memvalidasi hipotesis kami.

Sekarang kita akan mencoba melihat jumlah penumpang berdasarkan hari.

Catatan - 0 adalah awal minggu, yaitu 0 adalah Senin dan 6 adalah Minggu.

```
train.groupby('hari dalam seminggu')['Hitungan'].mean().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot di 0x7f2cfb5ac2b0>



Dari diagram batang di atas, kita dapat menyimpulkan bahwa jumlah penumpang pada hari Sabtu dan Minggu lebih sedikit dibandingkan hari-hari lainnya dalam seminggu. Sekarang kita akan melihat teknik dasar pemodelan. Sebelumnya kita akan menghilangkan variabel ID karena tidak ada hubungannya dengan jumlah penumpang.

```
kereta=kereta.drop('ID',1)
```

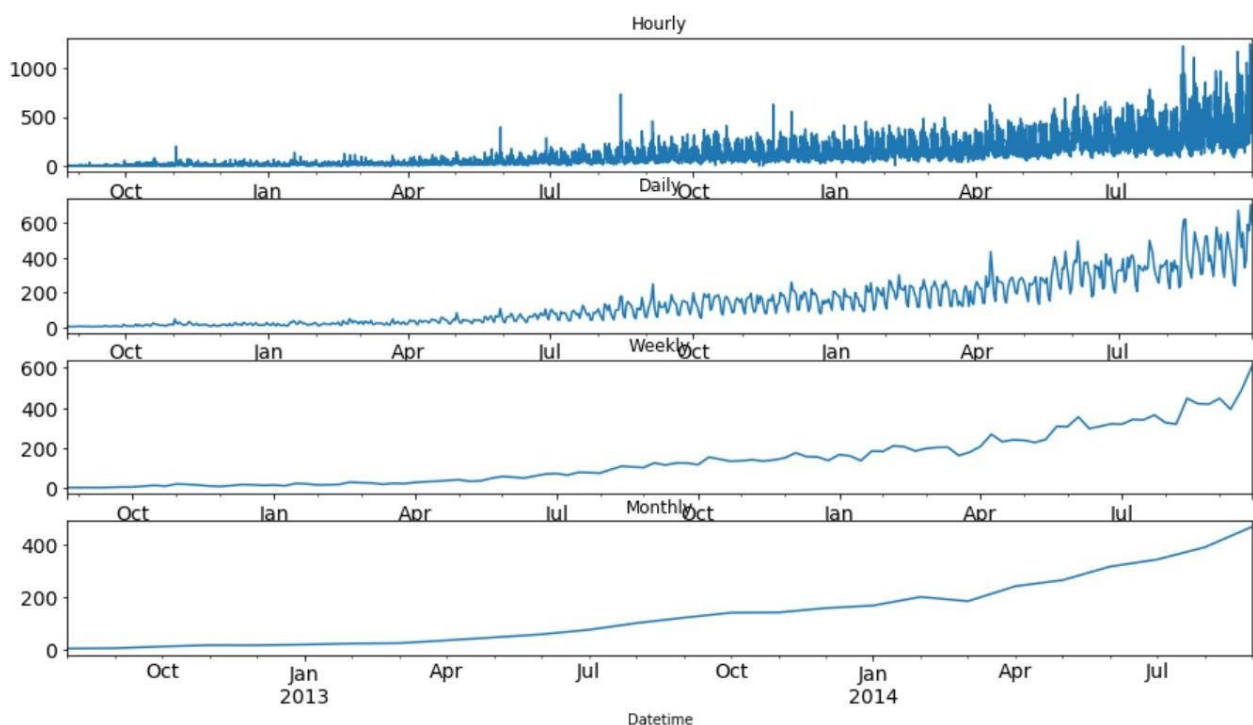
Seperti yang telah kita lihat bahwa ada banyak gangguan dalam rangkaian waktu per jam, kami akan menggabungkan rangkaian waktu per jam menjadi rangkaian waktu harian, mingguan, dan bulanan untuk mengurangi kebisingan dan membuatnya lebih stabil sehingga akan lebih mudah untuk model. untuk belajar.

```
train.Timestamp = pd.to_datetime(train.Datetime,format='%d-%m-%Y %H:%M') train.index =
train.Timestamp # Deret waktu
setiap jam setiap jam =
train.resample('H').berarti()
# Mengonversi ke rata-rata
harian setiap hari = train.resample('D').mean()
# Mengonversi ke mean mingguan
mingguan = train.resample('W').mean()
# Mengonversi ke rata-rata bulanan
bulanan = train.resample('M').mean()
```

Mari kita lihat rangkaian waktu per jam, harian, mingguan, dan bulanan.

```
fig, axs = plt.subplots(4,1)
setiap jam.Count.plot(figsize=(15,8), title= 'Setiap Jam', fontize=14, ax=axs[0]) setiap hari.Count.plot(figsize = (15,8),
title= 'Harian', ukuran font=14, ax=axs[1]) mingguan.Hitungan.plot(figsize=(15,8), title= 'Mingguan', ukuran font=14,
ax= axs[2]) bulanan.Count.plot(figsize=(15,8), title= 'Bulanan', fontize=14, ax=axs[3])
```

```
plt.tampilkan()
```



Kita dapat melihat bahwa deret waktu menjadi semakin stabil ketika kita menggabungkannya setiap hari, mingguan, dan bulanan.


Namun akan sulit untuk mengubah prediksi bulanan dan mingguan menjadi prediksi per jam, karena pertama-tama kita harus mengubah prediksi bulanan menjadi prediksi mingguan, mingguan menjadi harian, dan harian menjadi prediksi per jam, yang akan menjadi proses yang sangat diperluas. Jadi, kami akan mengerjakan rangkaian waktu harian.

```
test.Timestamp = pd.to_datetime(test.Datetime,format='%d-%m-%Y %H:%M') test.index  
= test.Timestamp
```

```
# Mengonversi ke tes rata-  
rata harian = test.resample('D').mean()
```

```
train.Timestamp = pd.to_datetime(train.Datetime,format='%d-%m-%Y %H:%M') train.index =  
train.Timestamp  
# Mengonversi ke rata-rata  
harian train = train.resample('D ').berarti()
```

Analisis Vidhya

 kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6134173-membagi-data-menjadi-pelatihan-dan-validasi-bagian

Karena kita telah memvalidasi semua hipotesis kita, mari kita lanjutkan dan membuat model untuk Peramalan Rangkaian Waktu. Namun sebelum kita melakukan itu, kita memerlukan dataset (validasi) untuk memeriksa performa dan kemampuan generalisasi model kita. Berikut adalah beberapa properti kumpulan data yang diperlukan untuk tujuan tersebut.

- Kumpulan data harus memiliki nilai sebenarnya dari variabel terikat yang dapat digunakan untuk memeriksa prediksi. Oleh karena itu, kumpulan data pengujian tidak dapat digunakan untuk tujuan.
- Model tidak boleh dilatih pada set data validasi. Oleh karena itu, kami tidak dapat melatih model pada kumpulan data kereta dan memvalidasinya juga.

Jadi, karena dua alasan di atas, biasanya kami membagi kumpulan data kereta menjadi dua bagian. Satu bagian digunakan untuk melatih model dan bagian lainnya digunakan sebagai dataset validasi. Sekarang ada beberapa cara untuk membagi kumpulan data kereta seperti Pembagian Acak, dll. Anda dapat mencari semua metode validasi yang berbeda

di sini: <https://www.analyticsvidhya.com/blog/2015/11/improve-model-Performance-cross-validation-in-python-r/>.

Untuk kursus ini, Kami akan menggunakan pembagian berdasarkan waktu yang dijelaskan di bawah.

1) Membagi data menjadi bagian pelatihan dan validasi

Sekarang kami akan membagi data kami dalam pelatihan dan validasi. Kami akan membuat model pada bagian kereta dan memprediksi pada bagian validasi untuk memeriksa keakuratan prediksi kami.

CATATAN - Merupakan praktik yang baik untuk membuat set validasi yang dapat digunakan untuk menilai model kami secara lokal. Jika metrik validasi (rmse) berubah sebanding dengan skor papan peringkat publik, ini berarti kami telah memilih teknik validasi yang stabil.

Untuk membagi data menjadi set pelatihan dan validasi, kami akan mengambil 3 bulan terakhir sebagai data validasi dan sisanya untuk data pelatihan. Kami hanya akan memakan waktu 3 bulan karena trennya paling banyak. Jika kami memerlukan waktu lebih dari 3 bulan untuk set validasi, set pelatihan kami akan memiliki poin data lebih sedikit karena total durasinya adalah 25 bulan. Jadi, sebaiknya luangkan waktu 3 bulan untuk set validasi.

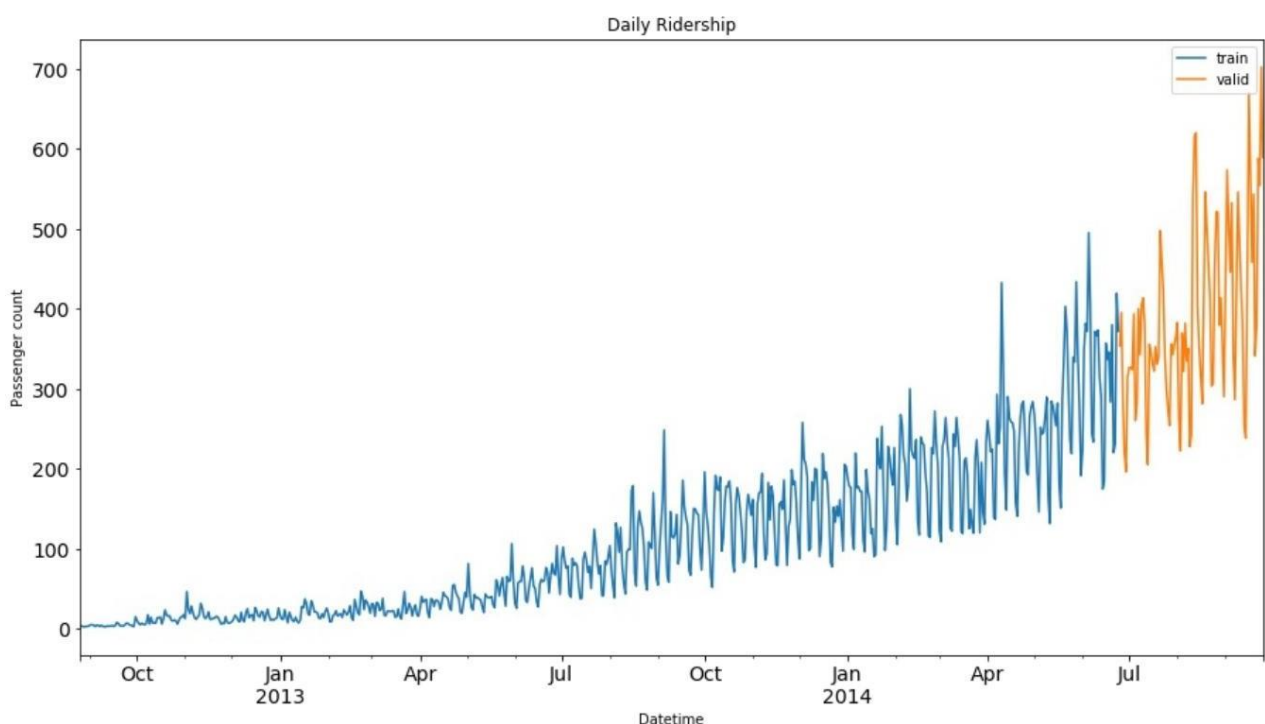
Tanggal awal kumpulan data adalah 25-08-2012 seperti yang telah kita lihat di bagian eksplorasi dan tanggal akhir adalah 25-09-2014.

Kereta=kereta.ix['25-08-2012':'24-06-2014'] valid=kereta.ix['25-06-2014':'25-09-2014']

- Kami telah melakukan validasi berbasis waktu di sini dengan memilih 3 bulan terakhir untuk data validasi dan sisanya pada data kereta. Jika kita melakukannya secara acak, ini mungkin berfungsi dengan baik untuk kumpulan data kereta tetapi tidak akan bekerja secara efektif pada kumpulan data validasi.
- Mari kita pahami seperti ini: Jika kita memilih pemisahan secara acak, maka akan diambil beberapa nilai dari awal dan beberapa dari tahun-tahun terakhir juga. Hal ini mirip dengan memprediksi nilai-nilai lama berdasarkan nilai-nilai masa depan yang tidak terjadi dalam skenario nyata. Jadi, pemisahan semacam ini digunakan saat mengerjakan masalah yang berhubungan dengan waktu.

Sekarang kita akan melihat bagaimana bagian pelatihan dan validasi dibagi.


```
Train.Count.plot(figsize=(15,8), title= 'Penumpang Harian', fontize=14, label='train')
valid.Count.plot(figsize=(15,8), title= 'Penumpang Harian ', fontize=14, label='valid')
plt.xlabel("Datetime") plt.ylabel("Jumlah penumpang") plt.legend(loc='best') plt.show()
```



Di sini bagian biru mewakili data kereta dan bagian oranye mewakili data validasi.

Kami akan memprediksi lalu lintas untuk bagian validasi dan kemudian memvisualisasikan seberapa akurat prediksi kami. Terakhir kami akan membuat prediksi untuk dataset pengujian.

Analisis Vidhya

 kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6134208-
teknik pemodelan

Kami akan melihat berbagai model sekarang untuk memperkirakan deret waktu . Metode yang akan kita lakukan yang dibahas untuk peramalan adalah:

i) Pendekatan Naif

ii) Rata-Rata Pergerakan

iii) Pemulusan Eksponensial Sederhana

iv) Model Tren Linier Holt

Kami akan membahas masing-masing metode ini secara rinci sekarang.

i) Pendekatan Naif

Dalam teknik peramalan ini, kita berasumsi bahwa titik harapan berikutnya sama dengan titik observasi terakhir. Jadi kita bisa mengharapkan garis lurus horizontal seperti prediksinya. Mari kita pahami dengan contoh dan gambar:

Misalkan kita mempunyai jumlah penumpang selama 5 hari seperti gambar di bawah ini:

1 10

2 12

3 14

4 13

5 15

Dan kita harus memprediksi jumlah penumpang untuk 2 hari ke depan. Pendekatan naif akan menetapkan jumlah penumpang hari ke-5 ke hari ke-6 dan ke-7, yaitu 15 akan ditetapkan ke hari ke-6 dan ke-7.

1 10

2 12

3 14

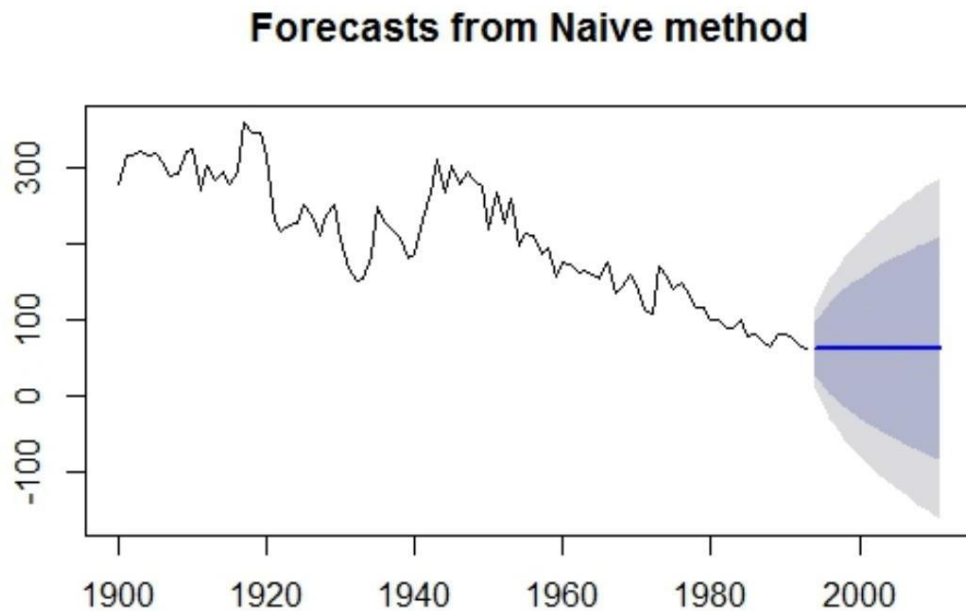
4 13

5 15

6 15

7 15

Sekarang mari kita pahami dengan sebuah contoh:

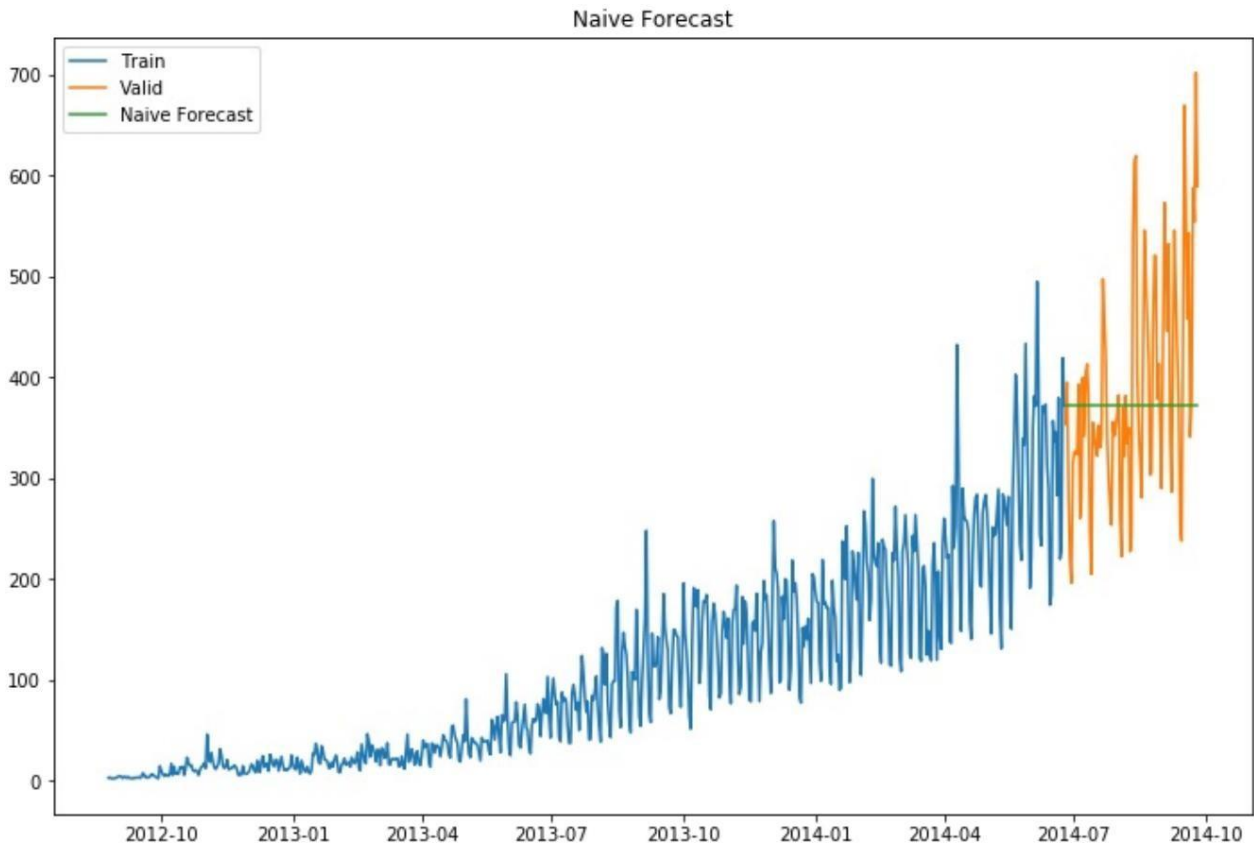


Contoh

Garis biru adalah prediksi di sini. Semua prediksi sama dengan titik pengamatan terakhir.

Mari membuat prediksi menggunakan pendekatan naif untuk set validasi.

```
dd= np.asarray(Train.Count)
y_hat = valid.copy()
y_hat['naif'] = dd[len(dd)-1]
plt.figure(figsize=(12,8))
plt.plot(Train .index, Kereta['Hitungan'], label='Kereta')
plt.plot(valid.index,valid['Hitungan'], label='Valid')
plt.plot(y_hat.index,y_hat['naif' ], label='Perkiraan Naif')
plt.legend(loc='terbaik')
plt.title("Perkiraan Naif")
plt.show()
```



- Kita dapat menghitung seberapa akurat prediksi kita menggunakan rmse (Root Mean Square Error).
- rmse
- adalah deviasi standar dari residu.
- Residual adalah ukuran seberapa jauh titik data dari garis regresi.
- Rumus rmse adalah:

$$rmse = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

Kami sekarang akan menghitung RMSE untuk memeriksa keakuratan model kami pada kumpulan data validasi.

```
dari sklearn.metrics impor mean_squared_error dari
impor matematika sqrt
rms = sqrt(mean_squared_error(valid.Count, y_hat.naive))
```

111.79050467496724

Kami dapat menyimpulkan bahwa metode ini tidak cocok untuk kumpulan data dengan variabilitas tinggi. Kita dapat mengurangi nilai perusahaan dengan mengadopsi teknik yang berbeda.

ii) Rata-Rata Pergerakan

Dalam teknik ini kami akan mengambil rata-rata jumlah penumpang selama beberapa waktu terakhir

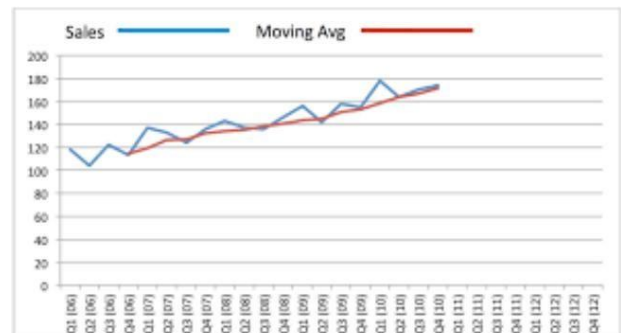
periode saja.

Mari kita ambil contoh untuk memahaminya:

Contoh

Di sini prediksi dibuat berdasarkan rata-rata beberapa poin terakhir, bukan berdasarkan semua nilai yang diketahui sebelumnya.

Mari kita coba rolling mean selama 10, 20, 50 hari terakhir dan visualisasikan hasilnya.

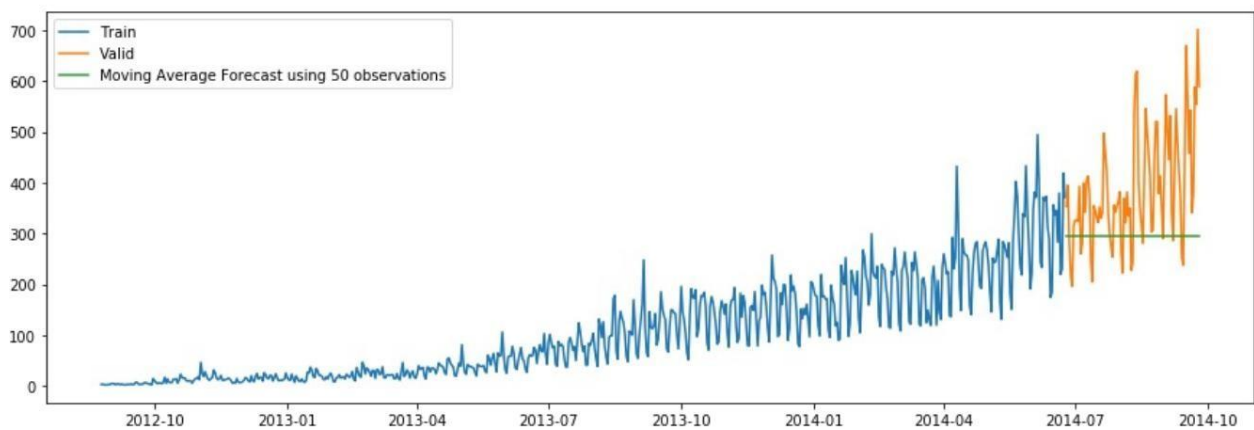
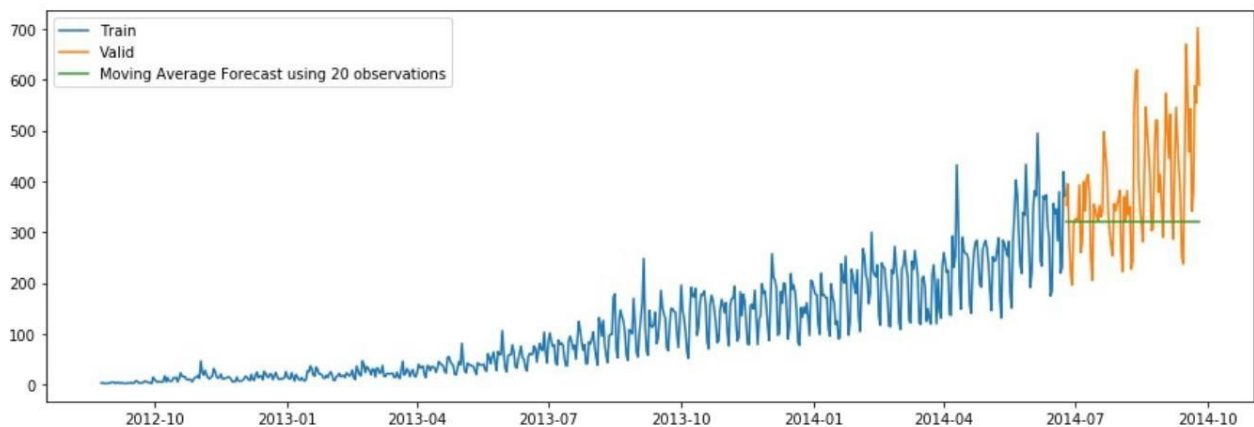
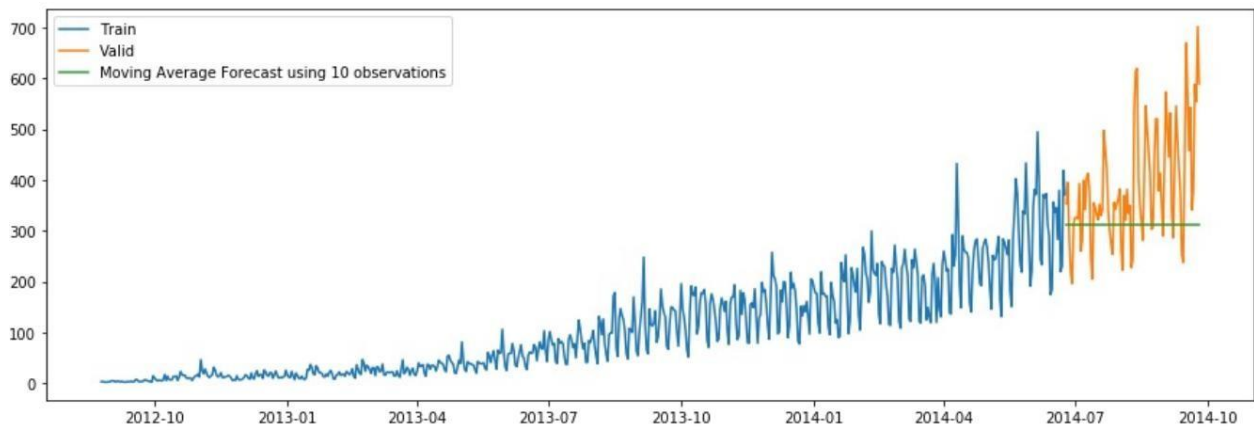


```
y_hat_avg = valid.copy()
y_hat_avg['moving_avg_forecast'] = Train['Count'].rolling(10).mean().iloc[-1] # rata-rata dari 10 pengamatan terakhir.

plt.gambar(figsize=(15,5))
plt.plot(Kereta['Hitungan'], label='Kereta')
plt.plot(valid['Hitungan'], label='Valid')
plt.plot(y_hat_avg['moving_avg_forecast'], label='Prakiraan Rata-Rata Pergerakan menggunakan 10 observasi') plt.legend(loc='best')
plt.show() y_hat_avg =
valid.copy()
y_hat_avg['moving_avg_forecast']
= Kereta['Count'].rolling(20).mean().iloc[-1] # rata-rata dari 20 observasi terakhir.

plt.gambar(figsize=(15,5))
plt.plot(Kereta['Hitungan'], label='Kereta')
plt.plot(valid['Hitungan'], label='Valid')
plt.plot(y_hat_avg['moving_avg_forecast'], label='Perkiraan Rata-Rata Pergerakan menggunakan 20 pengamatan') plt.legend(loc='best')
plt.show() y_hat_avg =
valid.copy()
y_hat_avg['moving_avg_forecast']
= Kereta['Count'].rolling(50).mean().iloc[-1] # rata-rata dari 50 observasi terakhir.

plt.gambar(figsize=(15,5))
plt.plot(Kereta['Hitungan'], label='Kereta')
plt.plot(valid['Hitungan'], label='Valid')
plt.plot(y_hat_avg['moving_avg_forecast'], label='Perkiraan Rata-Rata Pergerakan menggunakan 50 observasi') plt.legend(loc='best')
plt.show()
```



Kami mengambil rata-rata dari 10, 20, dan 50 observasi terakhir dan memperkirakannya berdasarkan data tersebut. Nilai ini dapat diubah pada kode di atas di bagian `.rolling().mean()`. Kita dapat melihat bahwa prediksi tersebut semakin lemah seiring dengan bertambahnya jumlah observasi.

```
rms = sqrt(mean_squared_error(valid.Hitungan, y_hat_avg.moving_avg_forecast))
```

144.19175679986802

iii) Pemulusan Eksponensial Sederhana

- Dalam teknik ini, kami memberikan bobot yang lebih besar pada pengamatan yang lebih baru daripada pengamatan di masa lalu.
- Bobotnya berkurang secara eksponensial seiring dengan pengamatan yang dilakukan jauh di masa lalu, bobot terkecil dikaitkan dengan pengamatan tertua.

CATATAN - Jika kita memberikan seluruh bobot pada nilai terakhir yang diamati saja, metode ini akan serupa dengan pendekatan naif. Jadi, kita dapat mengatakan bahwa pendekatan naif juga merupakan teknik pemulusan eksponensial sederhana di mana seluruh bobot diberikan pada nilai terakhir yang diamati.

Mari kita lihat contoh pemulusan eksponensial sederhana:

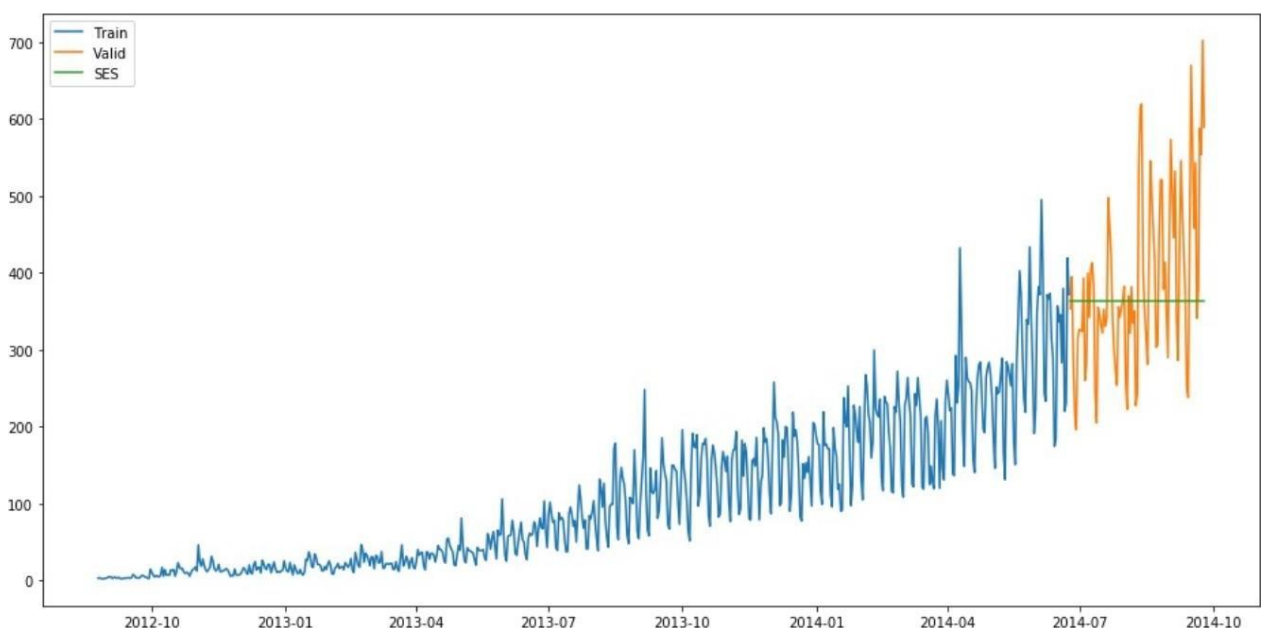
Contoh

Di sini prediksi dibuat dengan memberikan bobot lebih besar pada nilai terkini dan bobot lebih kecil pada nilai lama.

```
dari statsmodels.tsa.api impor ExponentialSmoothing, SimpleExpSmoothing,  
Holt y_hat_avg = valid.copy() fit2 =
```



```
SimpleExpSmoothing(np.asarray(Train['Count'])).fit(smoothing_level=0.6,optimized=False) y_hat_avg['SES'] =  
fit2.forecast(len(valid)) plt.figure(figsize=(16, 8))  
plt.plot(Kereta['Hitungan'],  
label='Kereta') plt.plot(valid['Hitungan'], label='Valid')  
plt.plot(y_hat_avg['SES'], label='SES')  
plt.legend(loc='terbaik') plt.show()
```



```
rms = sqrt(mean_squared_error(valid.Count, y_hat_avg.SES))
```

```
113.43708111884514
```

Kita dapat menyimpulkan bahwa kecocokan model telah meningkat seiring dengan berkurangnya nilai rmse.

iv) Model Tren Linier Holt

- Ini merupakan perpanjangan dari pemulusan eksponensial sederhana untuk memungkinkan perkiraan data dengan tren.
- Metode ini memperhitungkan tren kumpulan data. Fungsi perkiraan di metode ini merupakan fungsi dari level dan tren.

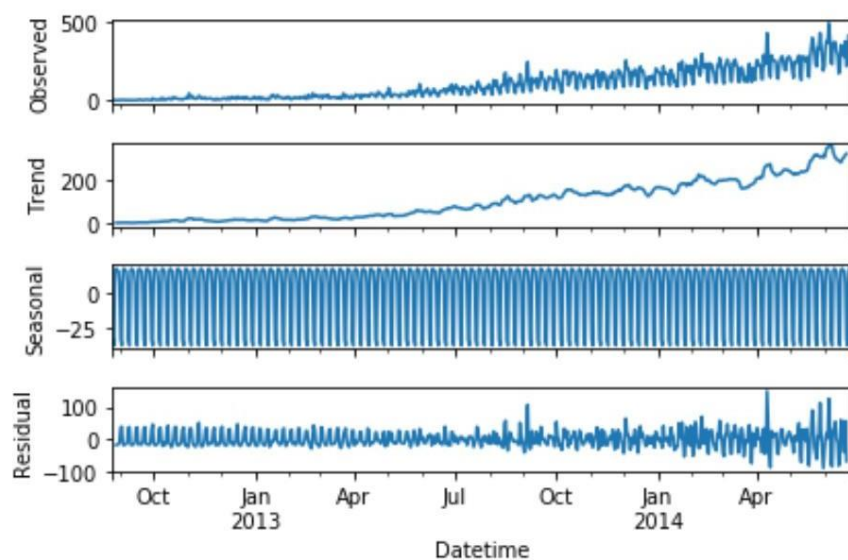
Pertama-tama mari kita visualisasikan tren, musiman, dan kesalahan dalam rangkaian tersebut.

Kita dapat menguraikan deret waktu menjadi empat bagian.

- Diamati, yang merupakan deret waktu asli.
- Tren, yang menunjukkan tren dalam deret waktu, yaitu perilaku kenaikan atau penurunan deret waktu.
- Musiman, yang memberitahu kita tentang kemusiman dalam rangkaian waktu.
- Residual, yang diperoleh dengan menghilangkan tren atau musiman apa pun dalam rangkaian waktu.

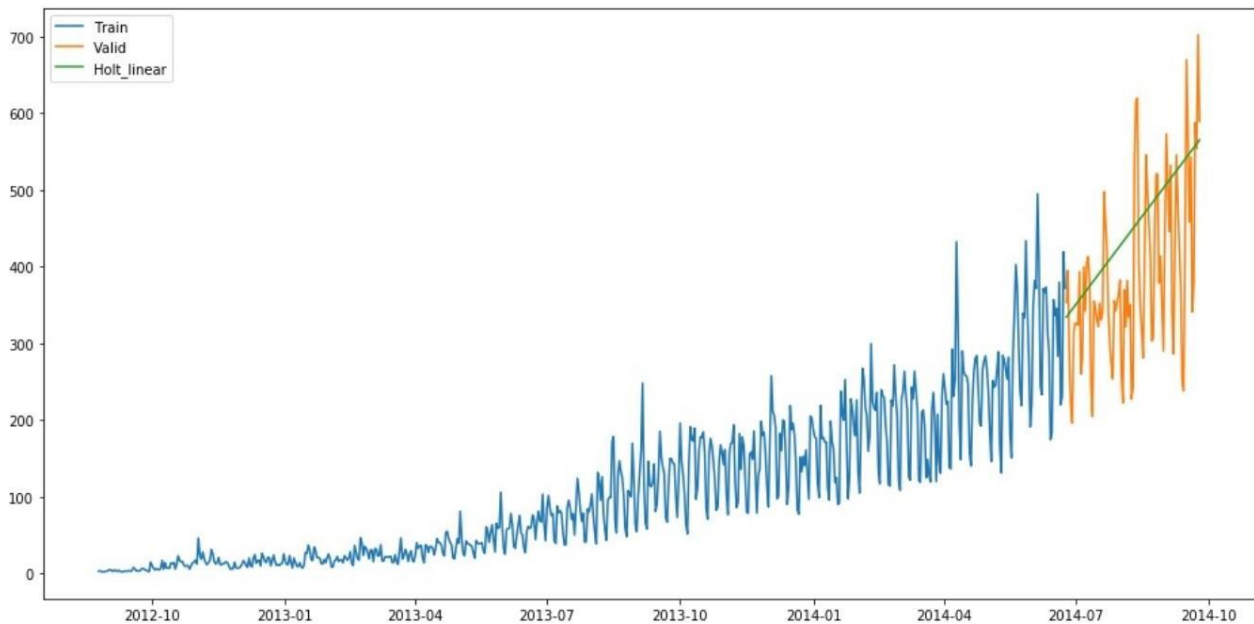
Mari kita visualisasikan semua bagian ini.

```
import statsmodels.api as sm
sm.tsa.seasonal_decompose(train.Count).plot()
hasil = sm.tsa.stattools.adfuller(train.Count)
plt.show()
```



Tren peningkatan terlihat pada dataset, maka sekarang kita akan membuat model berdasarkan tren tersebut.

```
y_hat_avg = valid.copy()
fit1 = Holt(np.asarray(Train['Count'])).fit(smoothing_level = 0.3,smoothing_slope = 0.1)
y_hat_avg['Holt_linear'] = fit1.forecast(len(valid))
plt.gambar(figsize=(16,8))
plt.plot(Kereta['Hitungan'], label='Kereta')
plt.plot(valid['Hitungan'], label='Valid')
plt.plot(y_hat_avg['Holt_linear'], label='Holt_linear')
plt.legend(loc='terbaik')
plt.show()
```



Kita dapat melihat garis miring di sini karena model telah mempertimbangkan tren rangkaian waktu.

Mari kita hitung nilai rmse model tersebut.

```
rms = sqrt(mean_squared_error(valid.Count, y_hat_avg.Holt_linear))
```

112.94278345314041

Dapat disimpulkan bahwa nilai rmse mengalami penurunan.

Sekarang kami akan memprediksi jumlah penumpang untuk kumpulan data pengujian menggunakan berbagai model.

3) Model Tren Linier Holt pada deret waktu harian

- Sekarang mari kita coba membuat model tren linier Holt pada deret waktu harian dan membuat prediksi pada kumpulan data pengujian.
- Kami akan membuat prediksi berdasarkan rangkaian waktu harian dan kemudian mendistribusikan prediksi harian tersebut ke prediksi per jam.
- Kami telah memasang model tren linier Holt pada kumpulan data kereta dan memvalidasinya menggunakan dataset validasi.

Sekarang mari kita memuat file penyerahan.

```
penyerahan=pd.read_csv("pengajuan.csv")
```

Kami hanya memerlukan ID dan Hitungan yang sesuai untuk penyerahan akhir.

Mari buat prediksi untuk kumpulan data pengujian.

```
prediksi=fit1.forecast(len(test))
```

Mari simpan prediksi ini dalam file pengujian di kolom baru.

```
tes['prediksi']=prediksi
```

Ingat ini adalah prediksi harian. Kami harus mengubah prediksi ini menjadi basis jam. * Untuk melakukannya pertama-tama kami akan menghitung rasio jumlah penumpang setiap jam setiap hari. * Kemudian kita akan mencari rasio rata-rata jumlah penumpang setiap jamnya dan kita akan mendapatkan 24 rasio. * Kemudian untuk menghitung prediksi per jam kita akan mengalikan prediksi harian dengan rasio per jam.


```

# Menghitung rasio hitungan per jam
train_original['ratio']=train_original['Count']/train_original['Count'].sum()

# Mengelompokkan rasio per
jam temp=train_original.groupby(['Hour'])['ratio'].sum()

# Groupby ke format csv
pd.DataFrame(temp, kolom=['Jam','rasio']).to_csv('GROUPby.csv')

temp2=pd.read_csv("GROUPby.csv")
temp2=temp2.drop('Jam.1',1)

# Gabungkan Tes dan tes_asli pada hari, bulan dan tahun
merge=pd.merge(test, test_original, on=('day','month','year'), how='left') merge['Hour']=
merge['Jam_y']
merge=merge.drop(['tahun','bulan','Tanggal_waktu','Jam_x','Jam_y'], sumbu=1)
# Memprediksi dengan menggabungkan
gabungan dan prediksi temp2=pd.merge(merge, temp2, on='Hour', how='left')

# Mengonversi rasio ke skala asli
prediksi['Hitungan']=prediksi['prediksi']*prediksi['ratio']*24 prediksi['ID']=prediksi['ID_y']

Mari hilangkan semua fitur lain dari file pengiriman dan pertahankan ID dan Hitung saja.


penyerahan=prediksi.drop(['ID_x','hari','ID_y','prediksi','Jam','rasio'],sumbu=1)
# Mengonversi penyerahan akhir ke format csv
pd.DataFrame(submission, Columns=['ID','Count']).to_csv('Holt linear.csv')

```

Model linier Holt memberikan nilai rmse 274.1596 di papan peringkat.

Sekarang mari kita lihat seberapa baik model musim dingin Holt dalam memprediksi jumlah penumpang untuk kumpulan data pengujian.

Analisis Vidhya

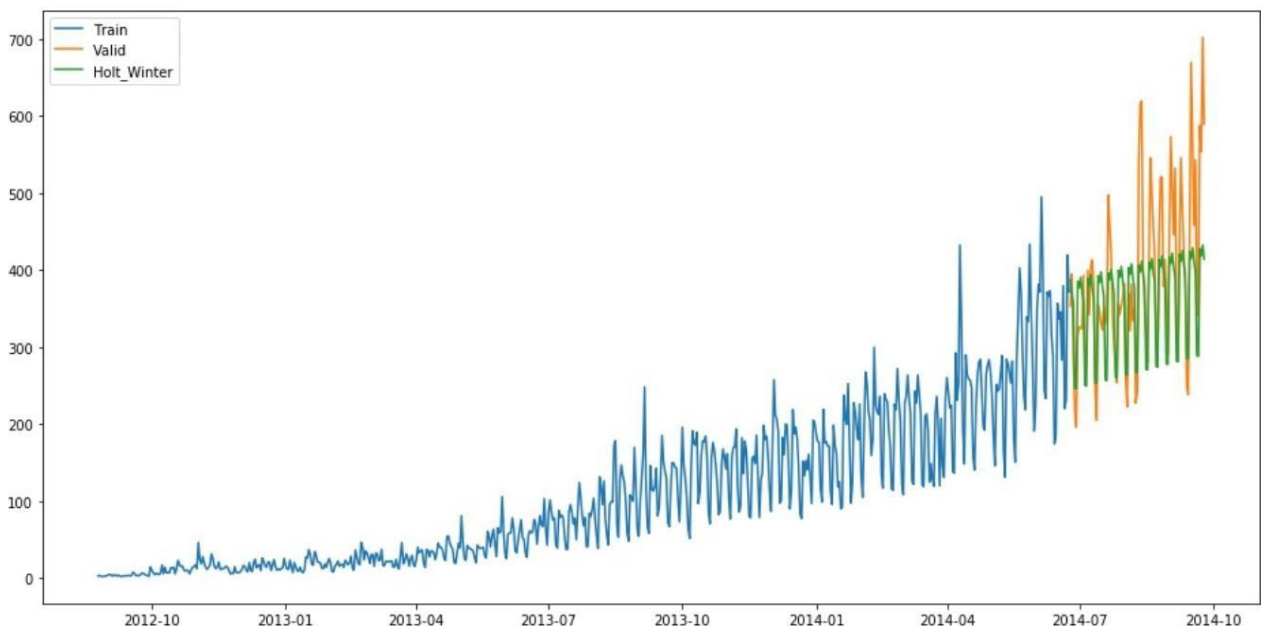
 kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6142664-holt-winter-s-model-on-daily-time-series

4) Model Holt Winter pada rangkaian waktu harian

- Kumpulan data yang menunjukkan serangkaian pola serupa setelah interval tetap dalam jangka waktu tertentu mengalami musiman.
- Model yang disebutkan di atas tidak memperhitungkan musiman kumpulan data saat melakukan perkiraan. Oleh karena itu diperlukan suatu metode yang memperhitungkan tren dan musiman untuk memperkirakan harga di masa depan.
- Salah satu algoritma yang dapat kita gunakan dalam skenario seperti itu adalah metode Holt's Winter. Ide di balik Holt's Winter adalah menerapkan pemulusan eksponensial pada komponen musiman selain level dan tren.

Pertama-tama, mari kita sesuaikan model pada kumpulan data pelatihan dan validasi menggunakan kumpulan data validasi.

```
y_hat_avg = valid.copy() fit1
= ExponentialSmoothing(np.asarray(Train['Count']), seasonal_periods=7 ,trend='add',seasonal='add',).fit()
y_hat_avg['Holt_Winter']
= fit1.forecast(len(valid)) plt.figure(figsize=(16,8))
plt.plot( Kereta['Hitungan'],
label='Kereta') plt.plot(valid['Hitungan'],
label='Valid') plt.plot(y_hat_avg['Holt_Winter'],
label='Holt_Winter') plt.legend(loc='terbaik') plt.show()
```



```
rms = sqrt(mean_squared_error(valid.Count, y_hat_avg.Holt_Winter)) print(rms)
```

82.37373991413227

Kita dapat melihat bahwa nilai rmse telah berkurang banyak dari metode ini. Mari kita perkirakan

Menghitung seluruh panjang kumpulan data Pengujian.

```
prediksi=fit1.forecast(len(test))
```

Sekarang kami akan mengubah jumlah penumpang harian ini menjadi jumlah penumpang per jam menggunakan pendekatan yang sama seperti yang kami ikuti di atas.

```
tes['prediksi']=prediksi
```

```
# Gabungkan Tes dan tes_asli pada hari, bulan dan tahun
```

```
merge=pd.merge(test, test_original, on=('day','month','year'), how='left') merge['Hour']=
```

```
merge['Jam_y']
```

```
merge=merge.drop(['tahun','bulan','Tanggal_waktu','Jam_x','Jam_y'], sumbu=1)
```

```
# Memprediksi dengan menggabungkan
```

```
gabungan dan prediksi temp2=pd.merge(merge, temp2, on='Hour', how='left')
```

```
# Mengonversi rasio ke skala asli
```

```
prediksi['Hitungan']=prediksi['prediksi']*prediksi['rasio']*24
```

Mari hilangkan semua fitur selain ID dan Hitungan

```
prediksi['ID']=prediksi['ID_y'] penyerahan=prediksi.drop(['hari','Jam','rasio','prediksi','ID_x','ID_y'],sumbu=1)
```

```
# Mengonversi penyerahan akhir ke format csv pd.DataFrame(pengajuan, kolom=
```

```
['ID','Count']).to_csv('Holt winters.csv')
```

- Model Holt Winters menghasilkan rmse 328.356 di papan peringkat.
- Kemungkinan alasan di balik hal ini mungkin karena model ini tidak begitu baik dalam memprediksi tren deret waktu, namun bekerja dengan sangat baik pada bagian musiman.

Hingga saat ini kami telah membuat model berbeda untuk tren dan musim. Tidak bisakah kita membuat model yang mempertimbangkan tren dan musim dari rangkaian waktu?

Ya, kami bisa. Kita akan melihat model ARIMA untuk peramalan deret waktu.

5) Pengenalan model ARIMA

- ARIMA adalah singkatan dari Auto Regression Integrated Moving Average. Ini ditentukan oleh tiga parameter terurut (p,d,q).
- Di sini p adalah urutan model autoregresif (jumlah jeda waktu) d adalah derajat perbedaan
- (berapa kali data dikurangi nilai masa lalu) q adalah urutan model rata-rata bergerak. Kami akan membahas
- lebih lanjut tentang parameter ini di bagian selanjutnya.

Peramalan ARIMA untuk deret waktu stasioner tidak lain hanyalah persamaan linier (seperti regresi linier).

Apa yang dimaksud dengan deret waktu stasioner?

Ada tiga kriteria dasar agar suatu deret dapat diklasifikasikan sebagai deret stasioner :

- Rata-rata deret waktu tidak boleh merupakan fungsi waktu. Itu harus konstan.
- Varians deret waktu tidak boleh merupakan fungsi waktu.
- Kovariansi suku ke-i dan suku (i+m) tidak boleh merupakan fungsi waktu.

Mengapa kita harus membuat deret waktu menjadi stasioner?

Kita membuat deret tersebut stasioner agar variabel-variabelnya independen. Variabel dapat bergantung dalam berbagai cara, namun hanya dapat independen dalam satu cara. Jadi, kita akan mendapat informasi lebih banyak ketika mereka sudah mandiri. Oleh karena itu deret waktu harus stasioner.

Jika deret waktu tidak stasioner, pertama-tama kita harus membuatnya stasioner. Untuk melakukan hal ini, kita perlu menghilangkan tren dan musiman dari data. Untuk mempelajari lebih lanjut tentang stasioneritas Anda dapat merujuk

[artikel ini: https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/](https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/)

6) Penyetelan parameter untuk model ARIMA

Pertama-tama kita harus memastikan bahwa deret waktu tersebut stasioner. Jika deret tersebut tidak stasioner maka akan dibuat stasioner.

Pemeriksaan Stasioneritas

- Kami menggunakan uji Dickey Fuller untuk memeriksa stasioneritas deret tersebut.
- Intuisi dibalik pengujian ini adalah untuk menentukan seberapa kuat deret waktu ditentukan oleh suatu tren.
- Hipotesis nol dari pengujian ini adalah bahwa deret waktu tidak stasioner (memiliki struktur yang bergantung pada waktu).
- Hipotesis alternatif (menolak hipotesis nol) adalah deret waktu tidak bergerak.

Hasil pengujian terdiri dari Statistik Uji dan beberapa Nilai Kritis untuk perbedaan tingkat kepercayaan. Jika 'Statistik Uji' lebih kecil dari 'Nilai Kritis', kita dapat menolak hipotesis nol dan mengatakan bahwa deret tersebut stasioner.

Kami menafsirkan hasil ini menggunakan Statistik Uji dan nilai kritis. Jika Uji Statistik lebih kecil dari nilai kritis, berarti kita menolak hipotesis nol (stasioner), sebaliknya Uji Statistik yang lebih besar menunjukkan kita menerima hipotesis nol (tidak stasioner).

Mari kita buat sebuah fungsi yang bisa kita gunakan untuk menghitung hasil tes Dickey-Fuller.

```

dari statsmodels.tsa.stattools impor adfuller def
test_stationarity (deretan waktu):
    #Menentukan statistik bergulir
    rolmean = pd.rolling_mean(timeseries, window=24) # 24 jam setiap hari rolstd =
    pd.rolling_std(timeseries, window=24)
    #Statistik pengguliran
    plot: orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean') std =
    plt. plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)
    #Lakukan tes Dickey-Fuller:

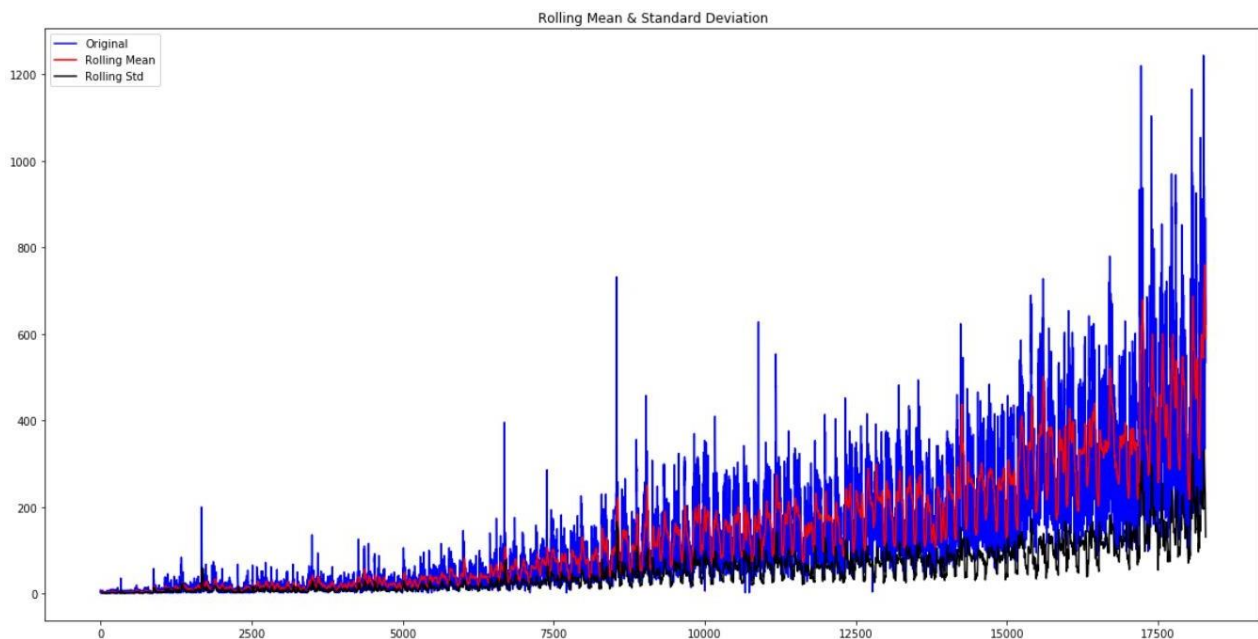
    dfctest = adfuller(timeseries, autolag='AIC')
    dfcoutput = pd.Series(dfctest[0:4], index=["Test Statistic",'p-value','#Lags Used','Jumlah of
Pengamatan yang Digunakan'])

    untuk kunci,nilai dalam
    dfctest[4].items(): dfcoutput['Nilai Kritis (%s)'%kunci] = nilai

dari matplotlib.pyplot import rcParams
rcParams['figure.figsize'] = 20,10

test_stationarity(kereta_asli['Hitungan'])

```



Hasil Uji Dickey-Fuller : Statistik Uji

-4.456561

p-value 0,000235

#Lag yang 45.000000

Digunakan Jumlah Pengamatan yang Digunakan 18242.000000

Nilai Kritis (1%) -3.430709

Nilai Kritis (5%) -2.861698

Nilai Kritis (10%) tipe d: -2.566854

float64

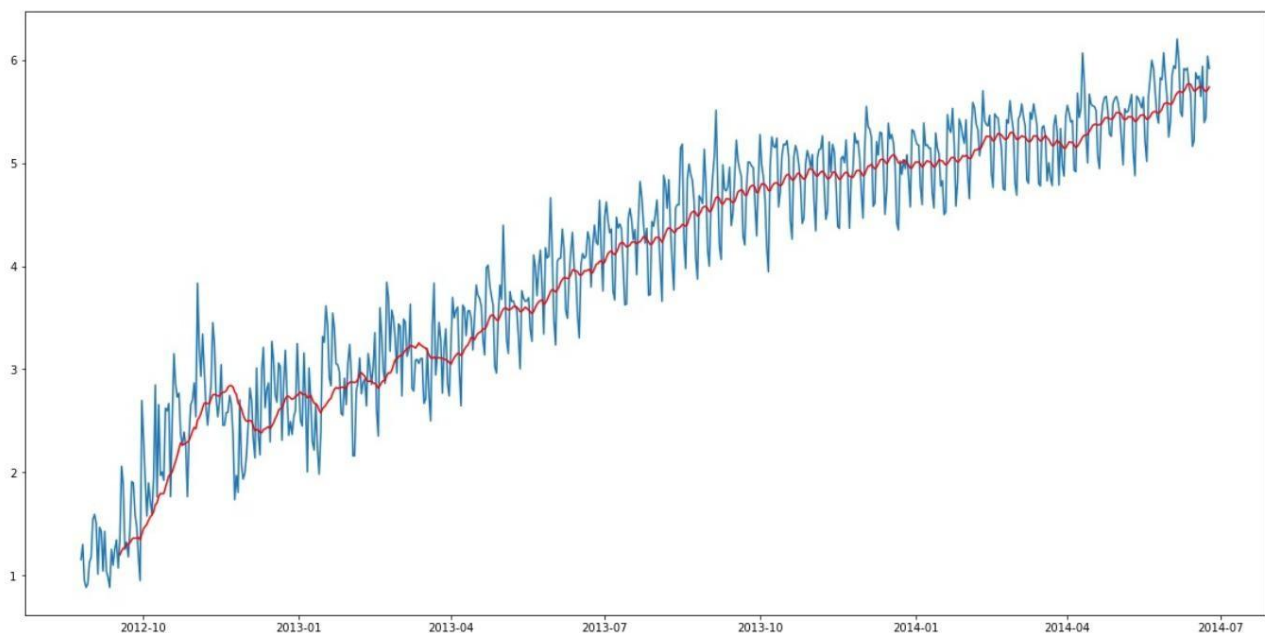
Statistik menunjukkan bahwa deret waktu stasioner karena Statistik Uji < Nilai kritis tetapi kita dapat melihat tren peningkatan pada data. Jadi, pertama-tama kita akan mencoba membuat datanya lebih stasioner. Untuk melakukan hal ini, kita perlu menghilangkan tren dan musiman dari data.

Menghapus Tren

- Tren terjadi ketika ada peningkatan atau penurunan data dalam jangka panjang. Tidak harus linier.
- Kami melihat tren peningkatan pada data sehingga kami dapat menerapkan transformasi yang memberikan penalti lebih besar pada nilai yang lebih tinggi dibandingkan nilai yang lebih kecil, misalnya transformasi log.
- Kami akan mengambil rata-rata bergulir di sini untuk menghilangkan tren. Kita akan mengambil ukuran jendela 24 berdasarkan fakta bahwa setiap hari memiliki 24 jam.

```
Train_log = np.log(Kereta['Hitungan']) valid_log  
= np.log(valid['Hitungan'])
```

```
moving_avg = pd.rolling_mean(Train_log, 24) plt.plot(Train_log)  
plt.plot(moving_avg,  
color = 'red') plt.show()
```

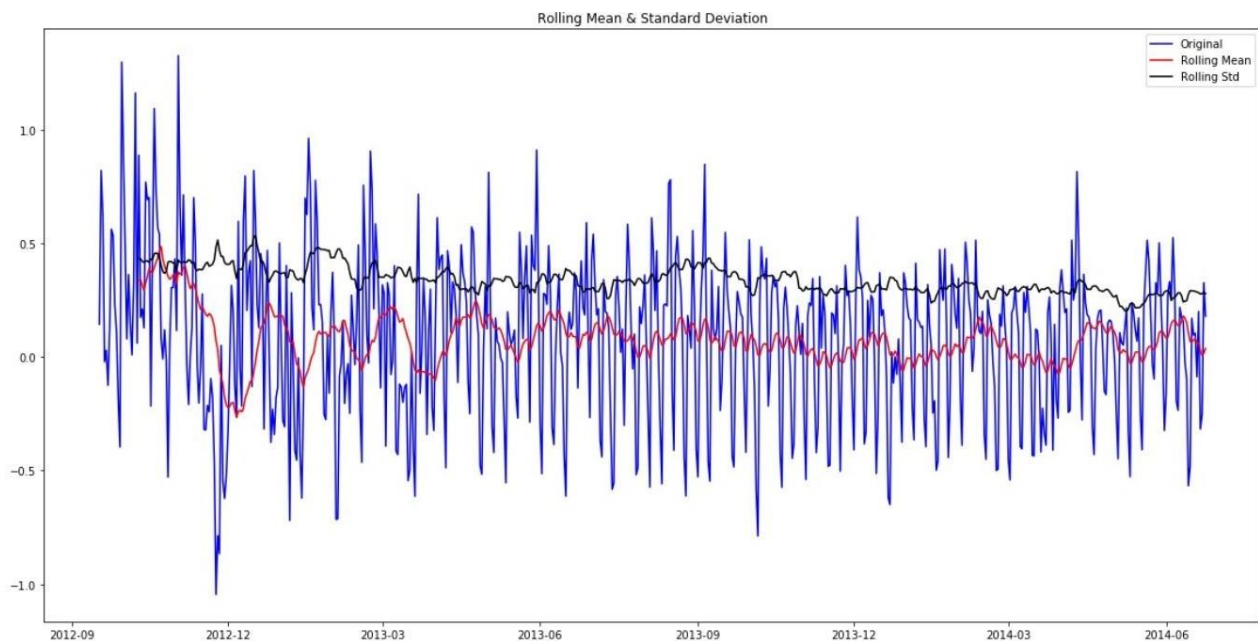


Jadi kita bisa mengamati tren peningkatannya. Sekarang kami akan menghilangkan tren peningkatan ini membuat deret waktu kita tidak bergerak.

```
train_log_moving_avg_diff = Kereta_log - rata-rata_bergerak
```

Karena kita mengambil rata-rata dari 24 nilai, rata-rata bergulir tidak ditentukan untuk 23 nilai pertama. Jadi mari kita hilangkan nilai nol itu.

```
train_log_moving_avg_diff.dropna(di tempat = Benar) test_stationarity(train_log_moving_avg_diff)
```



Hasil Tes Dickey-Fuller:

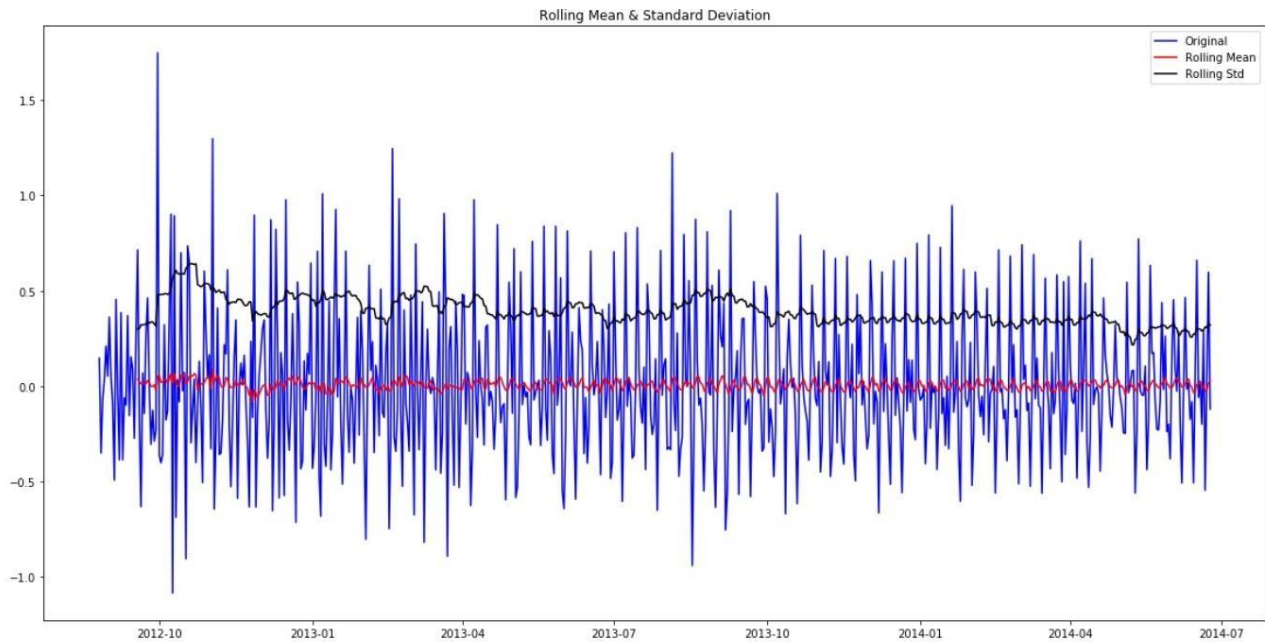
Statistik Uji	-5.861646e+00
p-value	3.399422e-07
#Lag yang	2.000000e+01
Digunakan Jumlah Pengamatan yang Digunakan	6.250000e+02
Nilai Kritis (1%)	-3.440856e+00
Nilai Kritis (5%)	-2.866175e+00
Nilai Kritis (10%) tipe d:	-2.569239e+00
float64	

Kita dapat melihat bahwa Statistik Uji sangat kecil dibandingkan dengan Nilai Kritis. Jadi, kita dapat yakin bahwa tren tersebut hampir hilang.

Sekarang mari kita menstabilkan mean deret waktu yang juga merupakan persyaratan untuk a deret waktu stasioner.

Perbedaan dapat membantu membuat rangkaian stabil dan menghilangkan tren.

```
train_log_diff = Train_log - Train_log.shift(1)
test_stationarity(train_log_diff.dropna())
```

Hasil Tes Dickey-Fuller:

Statistik Uji	-8.237568e+00
p-value	5.834049e-13
#Lag yang	1,900000e+01
Digunakan Jumlah Pengamatan yang Digunakan	6.480000e+02
Nilai Kritis (1%)	-3.440482e+00
Nilai Kritis (5%)	-2.866011e+00
Nilai Kritis (10%) tipe d:	-2.569151e+00
float64	

Sekarang kita akan menguraikan deret waktu menjadi tren dan musiman dan mendapatkan hasilnya sisa yang merupakan variasi acak dalam deret tersebut.

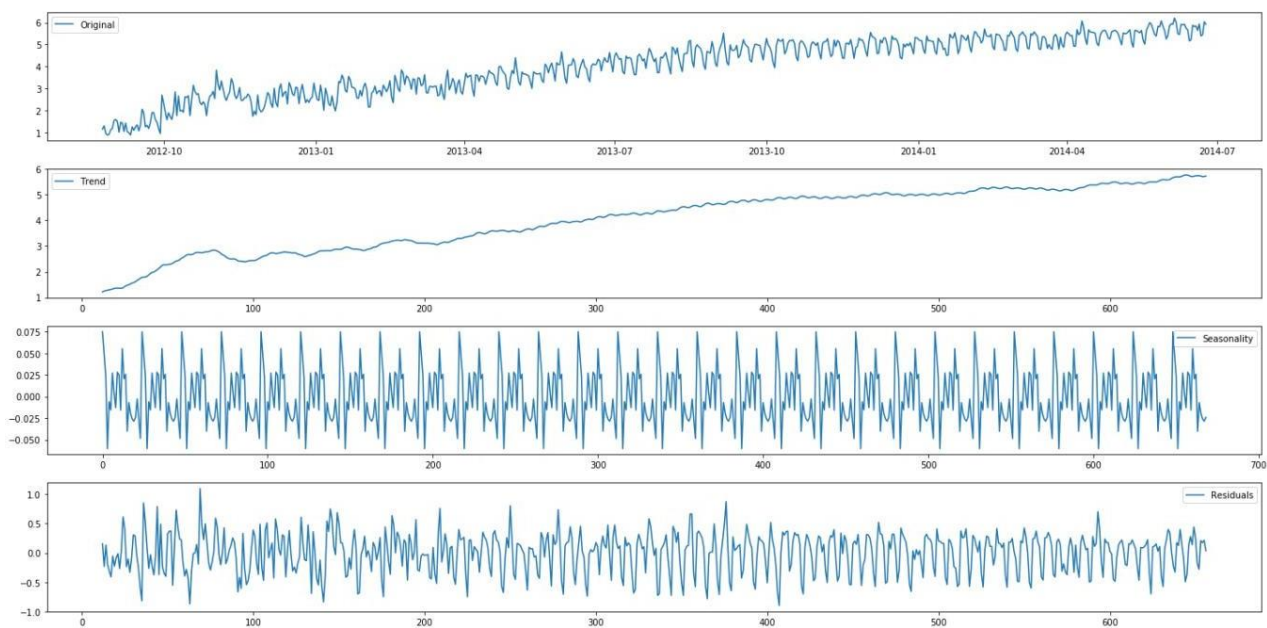
Menghapus Musiman

- Yang kami maksud dengan musim adalah fluktuasi periodik. Pola musiman terjadi ketika a seri dipengaruhi oleh faktor musiman (misalnya kuartal dalam setahun, bulan, atau hari dalam seminggu).
- Musiman selalu mempunyai periode yang tetap dan diketahui.
- Kita akan menggunakan dekomposisi musiman untuk menguraikan deret waktu menjadi tren, musiman dan residu.

```
dari statsmodels.tsa.seasonal import dekomposisi_musiman =
dekomposisi_musiman(pd.DataFrame(Train_log).Count.values, freq = 24)
```

```
tren = dekomposisi.tren
musiman = dekomposisi.sisa musiman
= dekomposisi.resid
```

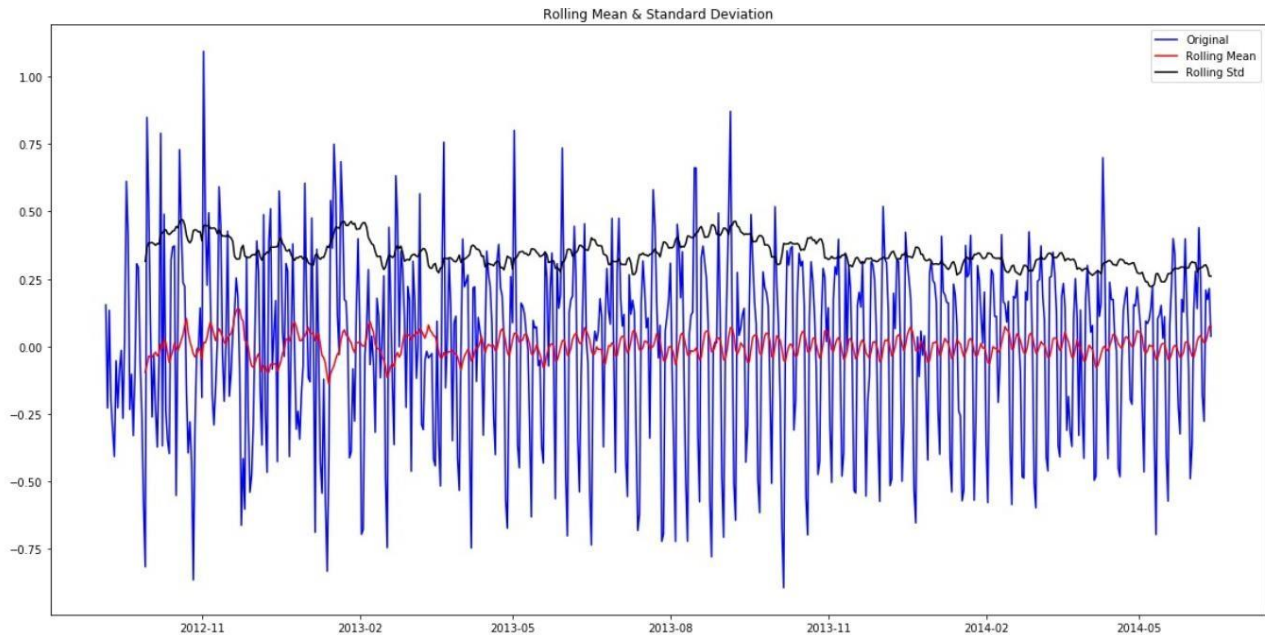
```
plt.subplot(411)
plt.plot(Train_log, label='Asli')
plt.legend(loc='terbaik')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend( loc='terbaik')
plt.subplot(413)
plt.plot(musiman,label='Kemusiman')
plt.legend(loc='terbaik')
plt.subplot(414)
plt.plot(sisa, label='Sisa ')
plt.legend(loc='terbaik')
plt.tight_layout()
plt.show()
```



Tren, residu, dan musiman dapat kita lihat dengan jelas pada grafik di atas.
Musiman menunjukkan tren yang konstan.

Mari kita periksa stasioneritas residu.

```
train_log_decompose = pd.DataFrame(residual)
train_log_decompose['date'] = Train_log.index
train_log_decompose.set_index('date', inplace = True) train_log_decompose.dropna(inplace=True)
test_stationarity(train_log_decompose[0])
```



Hasil Tes Dickey-Fuller:

Statistik Uji	-7.822096e+00
p-value	6.628321e-12
#Lag yang	2.000000e+01
Digunakan Jumlah Pengamatan yang Digunakan	6.240000e+02
Nilai Kritis (1%)	-3.440873e+00
Nilai Kritis (5%)	-2.866183e+00
Nilai Kritis (10%) tipe d:	-2.569243e+00
float64	

- Dari hasil tersebut dapat diartikan bahwa residunya stasioner.
- Sekarang kita akan memperkirakan deret waktu menggunakan model yang berbeda.

Peramalan deret waktu menggunakan ARIMA

- Pertama-tama kita akan menyesuaikan model ARIMA pada deret waktu kita yang harus kita temukan nilai yang dioptimalkan untuk parameter p, d, q .
- Untuk menemukan nilai optimal dari parameter ini, kita akan menggunakan ACF (Autocorrelation Fungsi) dan grafik PACF (Fungsi Autokorelasi Parsial).
- ACF adalah ukuran korelasi antara TimeSeries dengan versi yang tertinggal dari dirinya sendiri.
- PACF mengukur korelasi antara TimeSeries dengan versi yang tertinggal sendiri tetapi setelah menghilangkan variasi yang telah dijelaskan oleh intervensi tersebut perbandingan.

```

dari statsmodels.tsa.stattools impor acf, pacf lag_acf =
acf(train_log_diff.dropna(), nlags=25) lag_pacf =
pacf(train_log_diff.dropna(), nlags=25, method='ols')

```

Plot ACF dan PACF

```

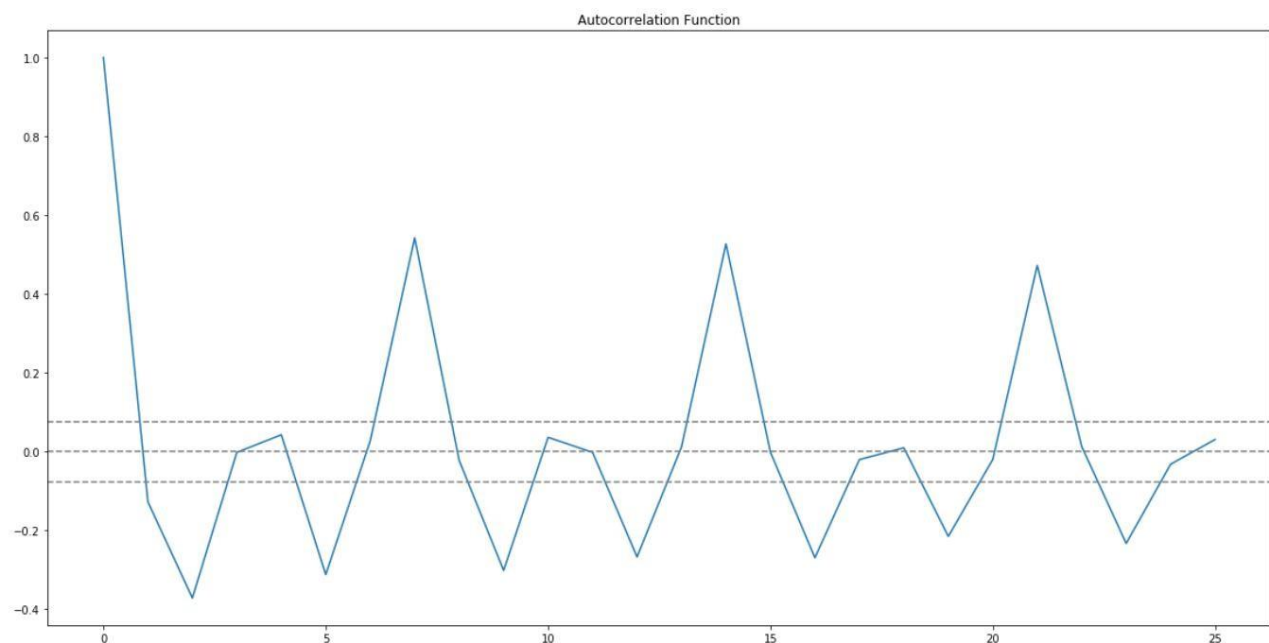
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray') plt.axhline(y=- 1.96/
np.sqrt(len(train_log_diff.dropna()))), linestyle='--',color='gray') plt.axhline(y=1.96/
np.sqrt(len(train_log_diff.dropna()))),linestyle='--',color='gray') plt. title('Fungsi Autokorelasi')
plt.show() plt.plot(lag_pacf)

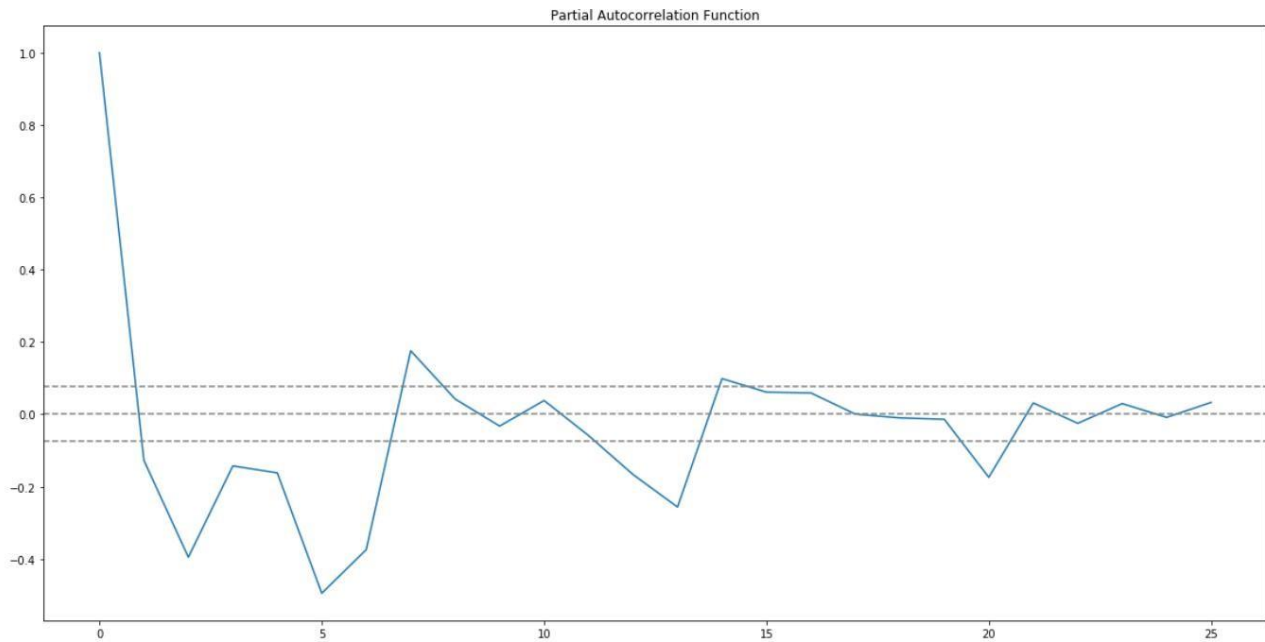
```

```

plt.axhline(y=0,linestyle='--',color='gray') plt.axhline(y=- 1.96/np.
sqrt(len(train_log_diff.dropna()))),linestyle='--',color='gray') plt.axhline(y=1.96/
np.sqrt(len(train_log_diff.dropna()))),linestyle=' --',color='abu-abu') plt.title('Fungsi Autokorelasi Parsial') plt.show()

```





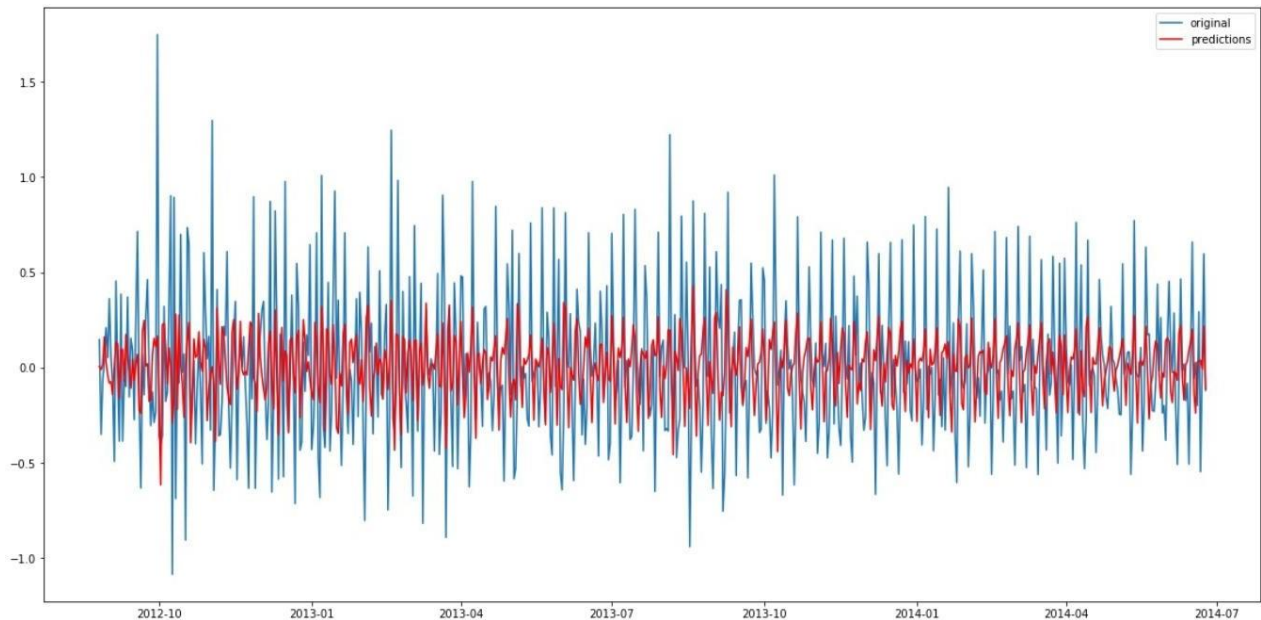
- Nilai p adalah nilai lag dimana grafik PACF melintasi interval kepercayaan atas untuk pertama kalinya. Dapat diperhatikan bahwa dalam kasus ini
- $p=1$. nilai q adalah nilai lag dimana grafik ACF melintasi interval kepercayaan atas untuk pertama kalinya. Dapat diperhatikan bahwa dalam kasus ini $q=1$.
- Sekarang kita akan membuat model ARIMA karena kita memiliki nilai p, q . Kita akan membuat model AR dan MA secara terpisah lalu menggabungkannya.

model AR

Model autoregresif menetapkan bahwa variabel keluaran bergantung secara linier pada nilai sebelumnya.

dari statsmodels.tsa.arima_model impor ARIMA

```
model = ARIMA(Train_log, order=(2, 1, 0)) # disini nilai q adalah nol karena hanya model AR
result_AR = model.fit(disp=-1)
plt.plot(train_log_diff.dropna(), label='asli')
plt.plot(result_AR.fittedvalues, color='merah', label='prediksi')
plt.legend(loc='terbaik')
plt.show()
```



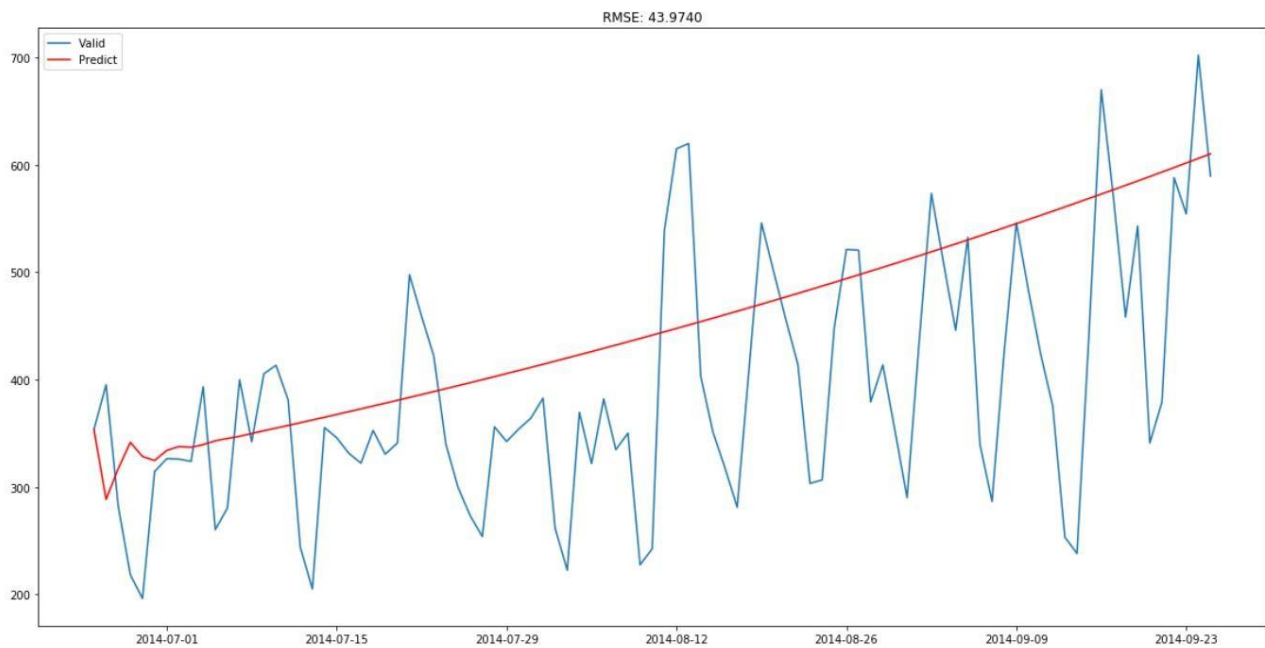
Mari kita gambarkan kurva validasi untuk model AR.

Kita harus mengubah skala model ke skala aslinya.

Langkah pertama adalah menyimpan hasil prediksi sebagai rangkaian terpisah dan mengamatinya.

```
AR_predict=results_AR.predict(start="25-06-2014", end="25-09-2014")
AR_predict=AR_predict.cumsum().shift().fillna(0) AR_predict1=pd.Series(np.ones(valid.shape[0]) *
np.log(valid['Count']))[0], indeks = valid.indeks)
AR_predict1=AR_predict1.add(AR_predict,fill_value=0)
AR_prediksi = np.exp(AR_predict1)

plt.plot(valid['Hitungan'], label = "Valid")
plt.plot(AR_predict, color = 'merah', label = "Prediksi")
plt.legend(loc= 'terbaik')
plt.title(' RMSE: %.4f'% (np.sqrt(np.dot(AR_predict, valid['Hitungan']))/valid.shape[0])) plt.show()
```



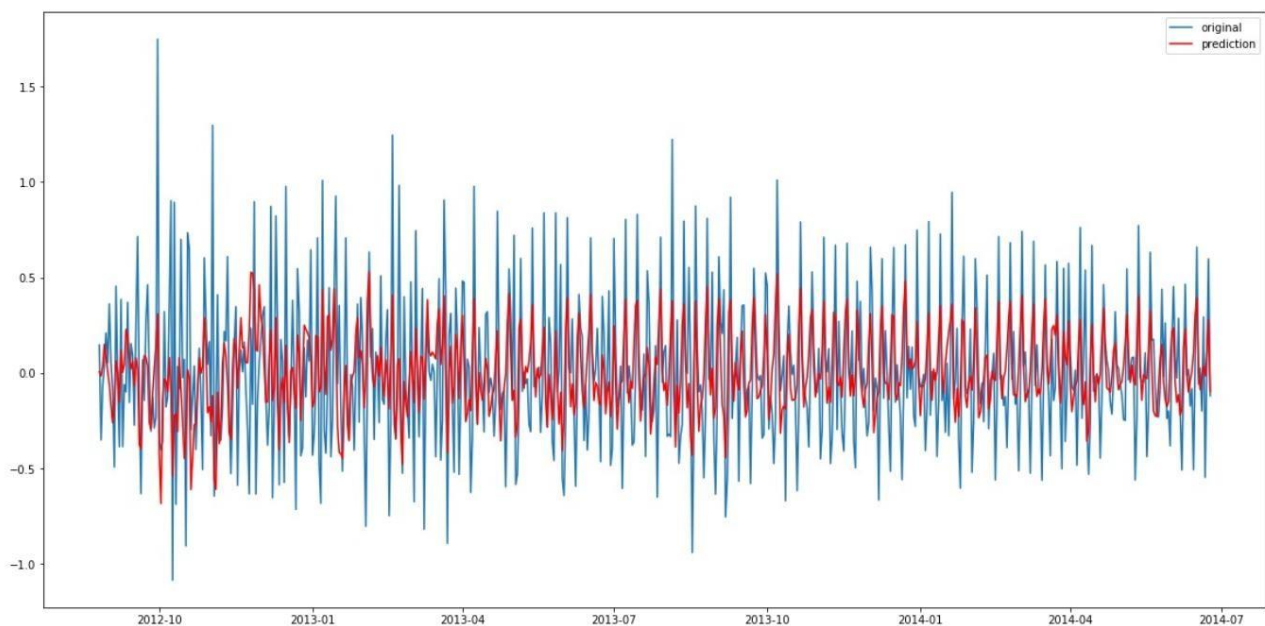
Di sini garis merah menunjukkan prediksi set validasi. Mari kita membangun model MA

Sekarang.

model MA

Model rata-rata bergerak menetapkan bahwa variabel keluaran bergantung secara linier pada nilai saat ini dan berbagai nilai masa lalu dari istilah stokastik (tidak dapat diprediksi secara tidak sempurna).

```
model = ARIMA(Train_log, order=(0, 1, 2)) # disini nilai p adalah nol karena hanya model MA
results_MA = model.fit(dispatch=-1)
plt.plot(train_log_diff.dropna(), label='asli')
plt.plot(results_MA.fittedvalues, color='merah', label='prediksi')
plt.legend(loc='terbaik')
plt.show()
```

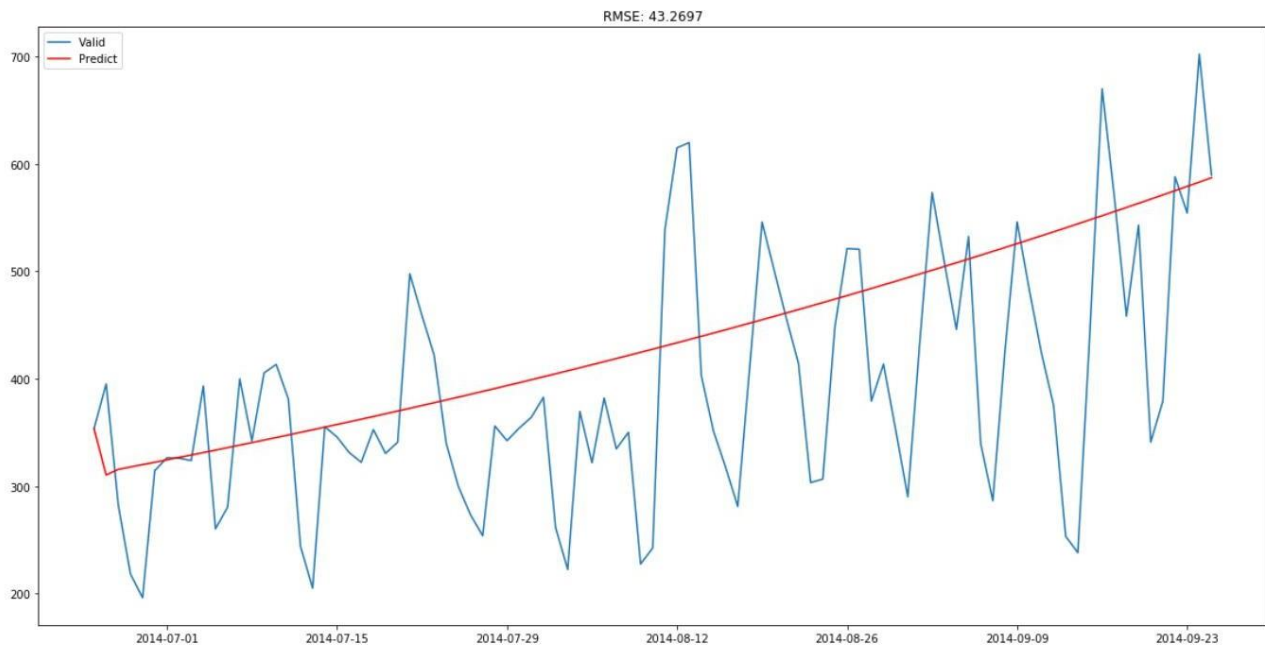



```

MA_predict=results_MA.predict(start="25-06-2014", end="25-09-2014")
MA_predict=MA_predict.cumsum().shift().fillna(0) MA_predict1=pd.Series(np.ones(valid.shape[0]) *
np.log(valid['Count']))[0], indeks = valid.indeks)
MA_predict1=MA_predict1.add(MA_predict,isi_nilai=0)
MA_prediksi = np.exp(MA_prediksi1)

plt.plot(valid['Hitungan'], label = "Valid")
plt.plot(MA_predict, color = 'merah', label = "Prediksi")
plt.legend(loc= 'terbaik')
plt.title(' RMSE: %.4f'% (np.sqrt(np.dot(MA_predict, valid['Hitungan']))/valid.shape[0])) plt.show()

```



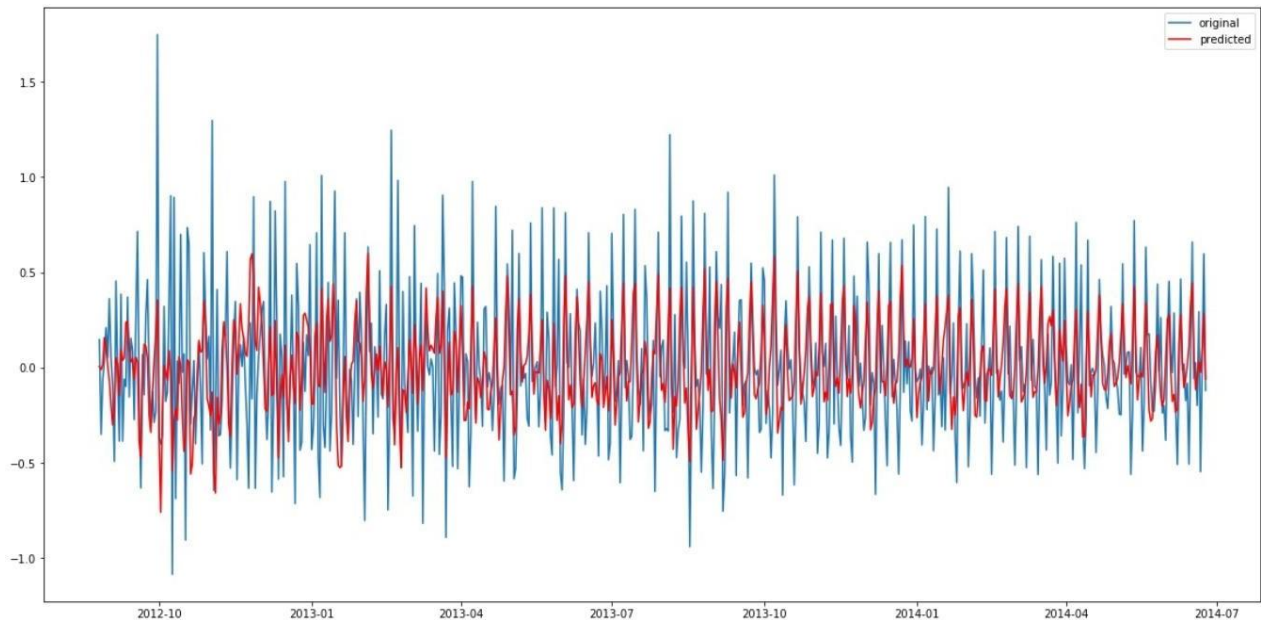
Sekarang mari kita gabungkan kedua model ini.

Model gabungan

```

model = ARIMA(Train_log, order=(2, 1, 2))
results_ARIMA = model.fit(dispatch=-1)
plt.plot(train_log_diff.dropna(), label='original')
plt.plot(results_ARIMA.fittedvalues, warna='merah', label='prediksi')
plt.legend(loc='terbaik')
plt.show()

```

Mari kita definisikan fungsi yang dapat digunakan untuk mengubah skala model ke skala aslinya.

```
def check_prediction_diff(predict_diff, diberikan_set):
    prediksi_diff= prediksi_diff.cumsum().shift().fillna(0)
    prediksi_base = pd.Series(np.ones(given_set.shape[0]) * np.log(given_set['Count'])[0], indeks =
diberikan_set.index)
    prediksi_log = prediksi_base.add(predict_diff,fill_value=0)
    prediksi = np.exp(predict_log)

    plt.plot(given_set['Count'], label = "Diberikan set")
    plt.plot(prediksi, warna = 'merah', label = "Prediksi")
    plt.legend(loc= 'terbaik')
    plt.title( 'RMSE: %.4f'% (np.sqrt(np.dot(prediksi, diberikan_set['Hitungan']))/given_set.shape[0]))
    plt.show()

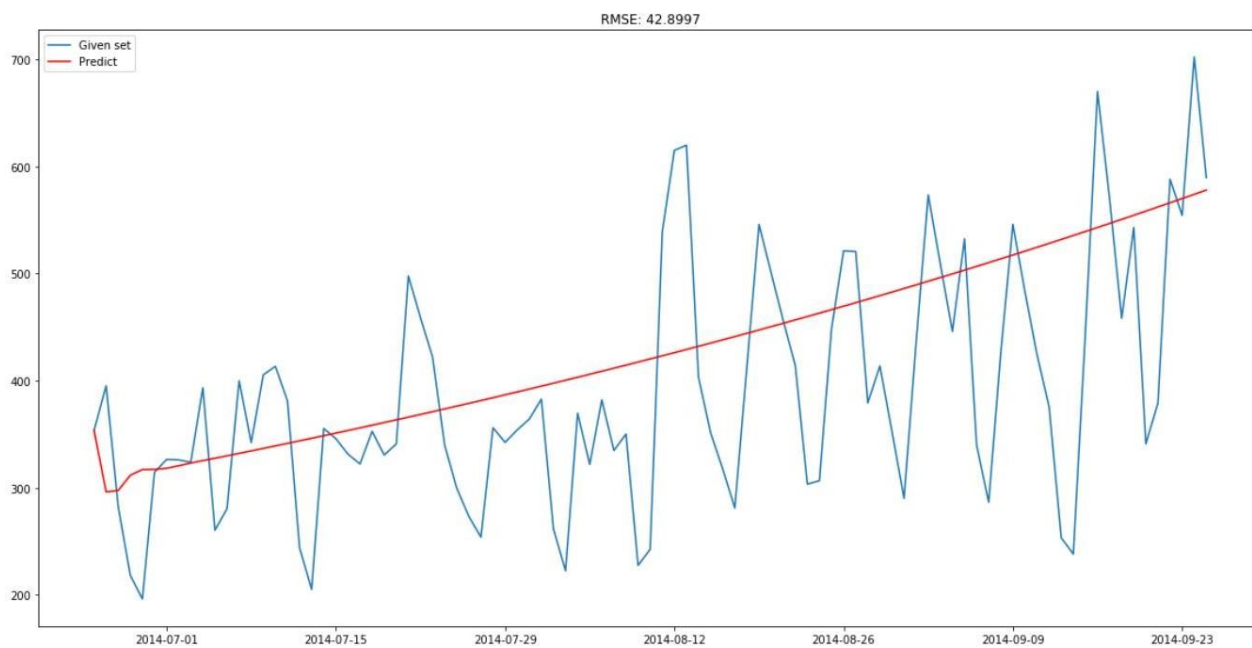
def check_prediction_log(predict_log, diberikan_set):
    prediksi = np.exp(predict_log)

    plt.plot(given_set['Count'], label = "Diberikan set")
    plt.plot(prediksi, warna = 'merah', label = "Prediksi")
    plt.legend(loc= 'terbaik')
    plt.title( 'RMSE: %.4f'% (np.sqrt(np.dot(prediksi, diberikan_set['Hitungan']))/given_set.shape[0]))
    plt.show()
```

Mari kita prediksi nilai set validasi.

```
ARIMA_predict_diff=results_ARIMA.predict(start="25-06-2014",    end="25-09-2014")

check_prediction_diff(ARIMA_predict_diff, valid)
```



Analisis Vidhya

kursus.analyticsvidhya.com/courses/take/creating-time-series-forecast-using-python/texts/6142826-sarimax-model-on-daily-time-series

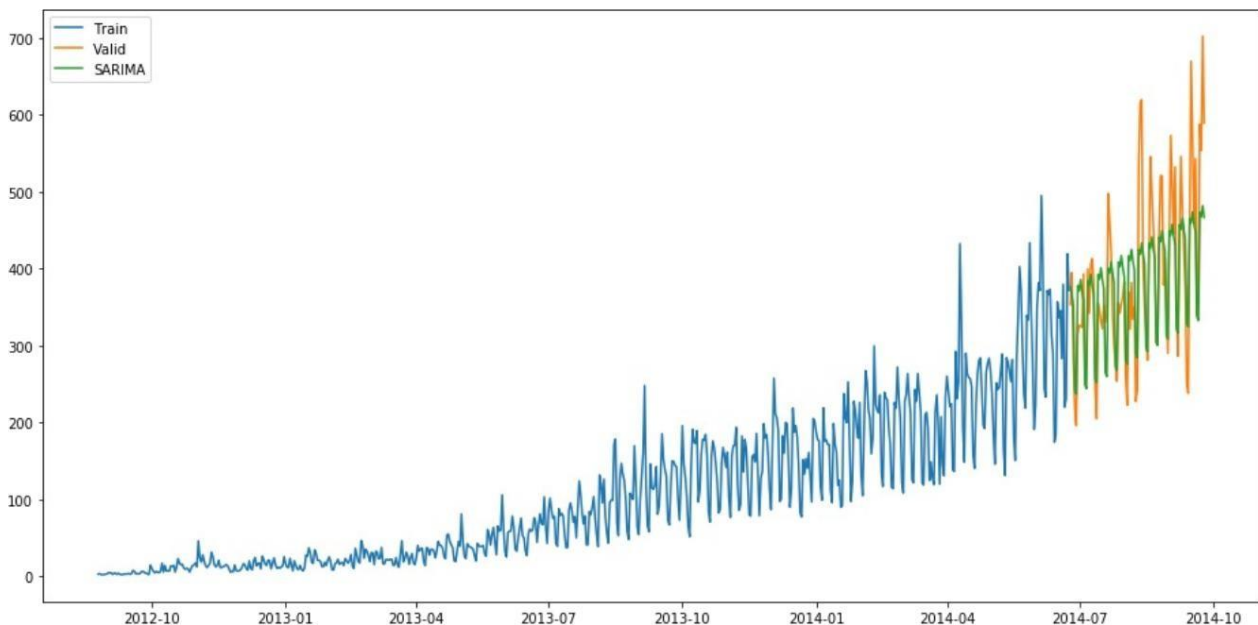
7) Model SARIMAX pada deret waktu harian

Model SARIMAX memperhitungkan musiman deret waktu. Jadi kita akan membangun model SARIMAX pada deret waktu.

```
import statsmodels.api sebagai sm
```

```
y_hat_avg = valid.copy() fit1  
= sm.tsa.statespace.SARIMAX(Train.Count, order=(2, 1, 4),seasonal_order=(0,1,1,7)).fit() y_hat_avg['  
SARIMA'] = fit1.predict(start="25-6-2014", end="25-9-2014", dinamis=Benar) plt.figure(figsize=(16,8))  
plt.plot( Kereta[ 'Hitungan'],  
label='Kereta') plt.plot(valid['Hitungan'],  
label='Valid') plt.plot(y_hat_avg['SARIMA'],  
label='SARIMA') plt.legend(loc ='terbaik') plt.show()
```

```
/home/pulkit/miniconda3/envs/av/lib/python3.6/site-packages/statsmodels-0.8.0-py3.6-linux-x86_64.egg/  
statsmodels/base/model.py:511: Peringatan Konvergensi: Maksimum Pengoptimalan kemungkinan gagal menyatu.  
Periksa mle_retvals  
"Periksa mle_retvals", Peringatan Konvergensi)
```



- Urutan dalam model di atas mewakili urutan model autoregresif (jumlah jeda waktu), tingkat perbedaan (berapa kali nilai masa lalu data dikurangi) dan urutan model rata-rata bergerak.

- Urutan musiman mewakili urutan komponen musiman model untuk parameter AR, perbedaan, parameter MA, dan periodisitas.
- Dalam kasus kami, periodisitasnya adalah 7 karena ini adalah rangkaian waktu harian dan akan berulang setiap 7 hari.

Mari kita periksa nilai rmse untuk bagian validasi.

```
rms = sqrt(mean_squared_error(valid.Count, y_hat_avg.SARIMA))
```

```
69.70093730473587
```

Sekarang kami akan memperkirakan rangkaian waktu untuk data Pengujian yang dimulai dari 26-9-2014 dan berakhir pada 26-4-2015.

```
prediksi=fit1.prediksi(mulai="26-9-2014", akhir="26-4-2015", dinamis=Benar)
```

Perhatikan bahwa ini adalah prediksi harian dan kami memerlukan prediksi setiap jam. Jadi, prediksi harian ini akan kami distribusikan ke dalam hitungan per jam. Untuk melakukannya, kami akan mengambil rasio distribusi jumlah penumpang per jam dari data kereta api dan kemudian kami akan mendistribusikan prediksi dalam rasio yang sama.

```
tes['prediksi']=prediksi
```

```
# Gabungkan Tes dan tes_asli pada hari, bulan dan tahun
merge=pd.merge(test, test_original, on=('day','month','year'), how='left')
merge['Hour']=merge['Jam_y']
merge=merge.drop(['tahun','bulan','Tanggal_waktu','Jam_x','Jam_y'], sumbu=1)
```

```
# Memprediksi dengan menggabungkan
gabungan dan prediksi temp2=pd.merge(merge, temp2, on='Hour', how='left')
```

```
# Mengonversi rasio ke skala asli
prediksi['Hitungan']=prediksi['prediksi']*prediksi['rasio']*24
```

Mari kita hilangkan semua variabel selain ID dan Hitung

```
prediksi['ID']=prediksi['ID_y']
penyerahan=prediksi.drop(['hari','Jam','rasio','prediksi','ID_x','ID_y'],sumbu=1)
```

```
# Mengonversi penyerahan akhir ke format csv
pd.DataFrame(submission, Columns=['ID','Count']).to_csv('SARIMAX.csv')
```

Metode ini memberi kami skor rmse paling sedikit. Nilai rmse di papan peringkat adalah 219.095.

Apa lagi yang bisa dicoba untuk meningkatkan model Anda lebih lanjut?

- Anda dapat mencoba membuat rangkaian waktu mingguan dan membuat prediksi untuk rangkaian tersebut, lalu mendistribusikan prediksi tersebut ke dalam prediksi harian dan kemudian per jam.
- Gunakan kombinasi model (ensemble) untuk mengurangi rmse. Untuk membaca lebih lanjut tentang teknik ansambel Anda dapat merujuk artikel berikut:

- <https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/>
- <https://www.analyticsvidhya.com/blog/2015/09/questions-ensemble-modeling/>
- Untuk membaca lebih lanjut mengenai analisis deret waktu, Anda dapat merujuk pada artikel berikut:
- <https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/>
- <https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/>