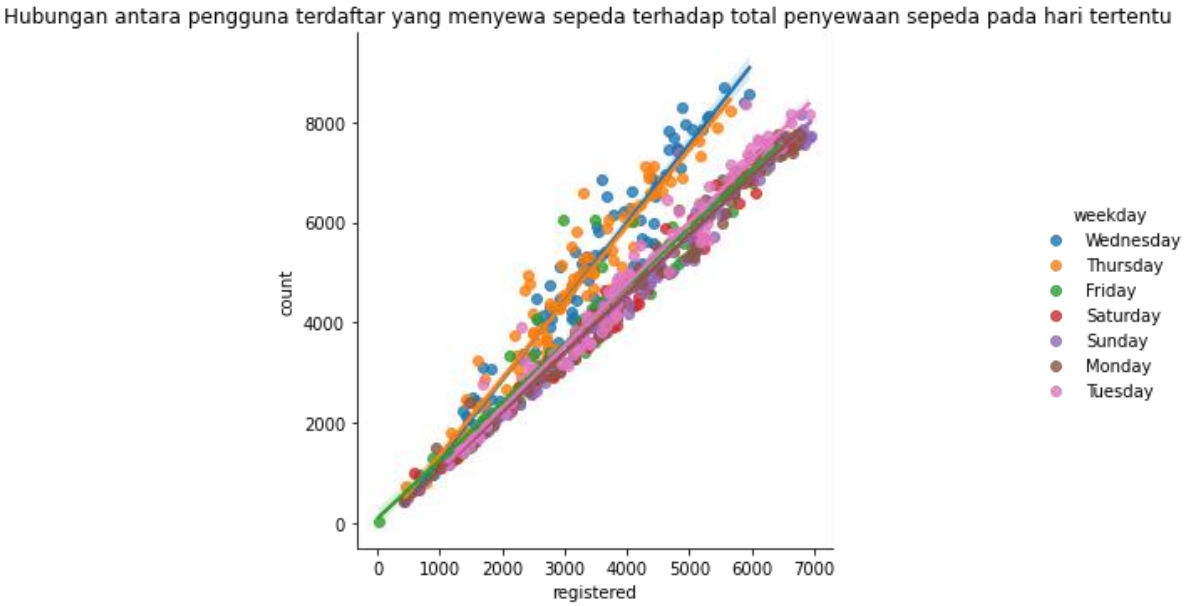
I. Pendahuluan

Dataset ini merupakan data kumpulan data projek berbagi sepeda di mana proses persewaan dan pengambilan sangat otomatis dan fleksibel dimana sepeda dapat disewa di satu lokasi dan dikembalikan di lokasi lain tanpa harus berurusan dengan manusia. Dengan dataset ni saya sebagai seorang analis data akan memprediksi tingkat persewaan sepeda harian dari variabel lingkungan dan musim. Asumsi yang saya gunakan adalah variabel *registered* adalah bagian dari target (*Count*) sehingga tidak saya masukkan dalam pemodelan. Hipotesis awal saya adalah variable lingkungan dan musim berpengaruh terhadap jumlah pnyewaan sepeda.

II. Explanatory Data Analysis (EDA)

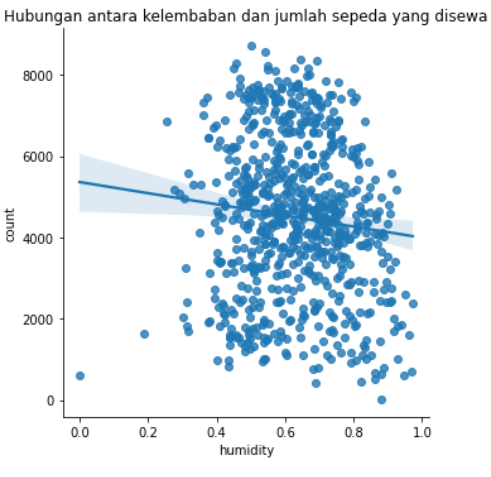
- **Hubungan antara jumlah sepeda yang disewa oleh pengguna terdaftar yang menyewa sepeda terhadap total penyewaan sepeda pada hari tertentu**

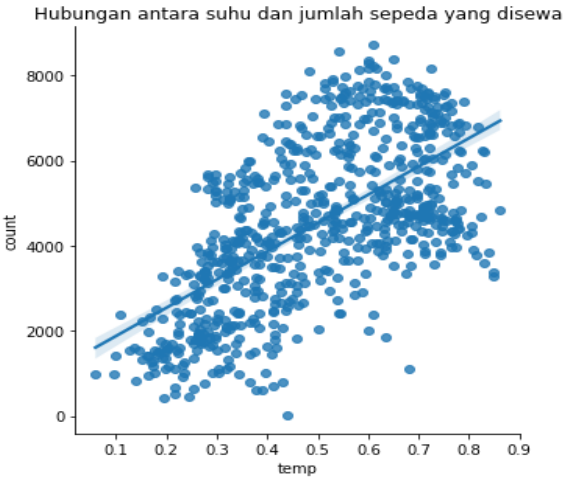


- Dari gradien yang lebih curam pada hari rabu dan kamis tersebut sehingga diketahui bahwa terdapat total penyewaan sepeda lebih tinggi daripada jumlah penyewaan sepeda oleh pengguna terdaftar pada hari tertentu.
- Hal ini menandakan bahwa ada kenaikan total penyewaan sepeda pada hari tertentu

- **Apa hubungan antara suhu dan jumlah sepeda yang disewa? Apa hubungan antara kelembaban dan jumlah sepeda yang disewa?**

Visualisasi :

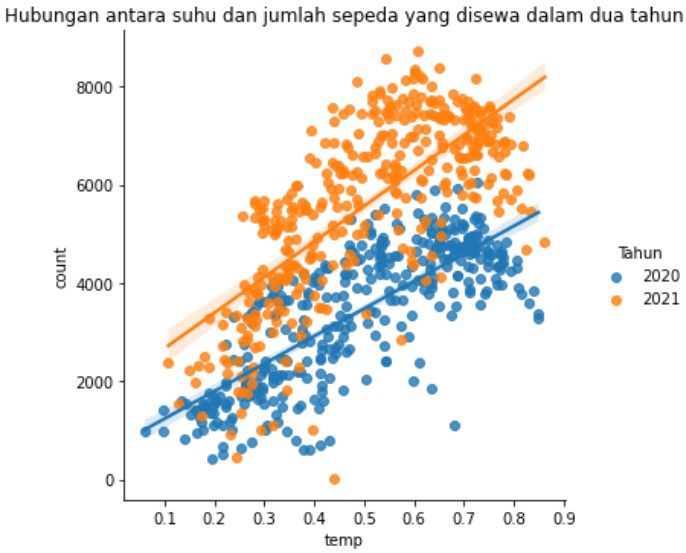




- Jumlah sepeda akan meningkat seiring dengan meningkatnya temperatur sampai titik optimum dan akan menurun jumlah sepeda yang dijual jika temperatur titik dinaikkan setelah titik optimum tersebut.
- Pada Humidity vs count, terdapat hubungan yang berbanding terbalik.

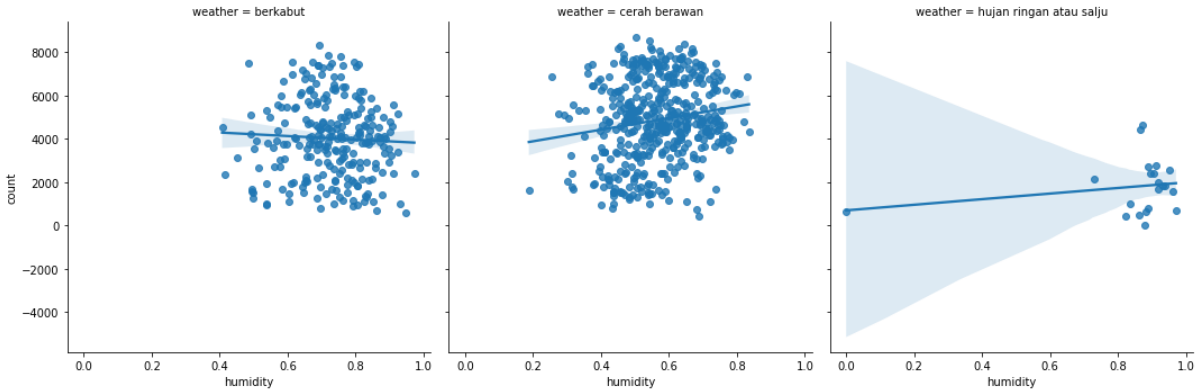
➢ **Apakah hubungan antara suhu dan jumlah sepeda yang disewa sama dalam dua tahun?**

Visualisasi :



- Dari grafik Scatter bisa dilihat bahwa hubungan antara suhu dengan jumlah sepeda adalah sama Dalam 2 tahun tersebut yaitu jumlah sepeda akan meningkat seiring dengan meningkatnya temperatur sampai titik optimum dan akan menurun jumlah sepeda yang dijual jika temperatur titik dinaikkan setelah titik optimum tersebut.
- Dapat dilihat juga bahwa jumlah penyewaan sepeda pada 2021 akan lebih banyak daripada tahun 2020

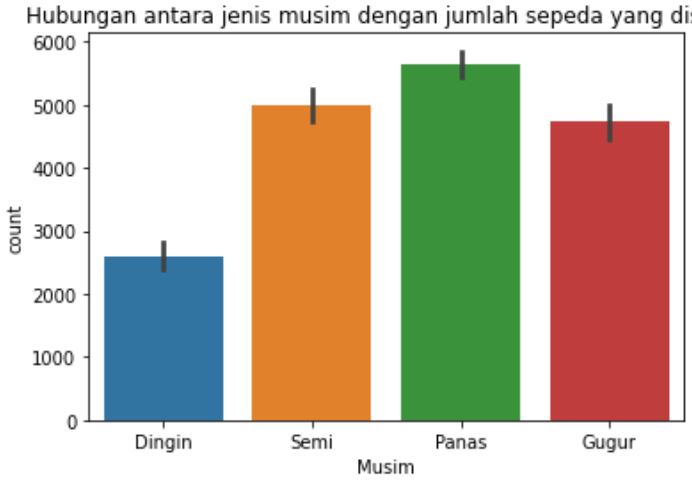
➢ **Apakah hubungan antara kelembaban dan jumlah penyewaan sepeda akan berbeda dalam kondisi cuaca yang berbeda?**



- Dari grafik diatas dapat diketahui bahwa pada cuaca berkabut hubungan kelembaban dan jumlah sepeda yang disewa adalah berbanding terbalik
- Namun sebaliknya pada cuaca cerah berawan hubungan kelembaban dan jumlah sepeda yang disewa adalah berbanding lurus.

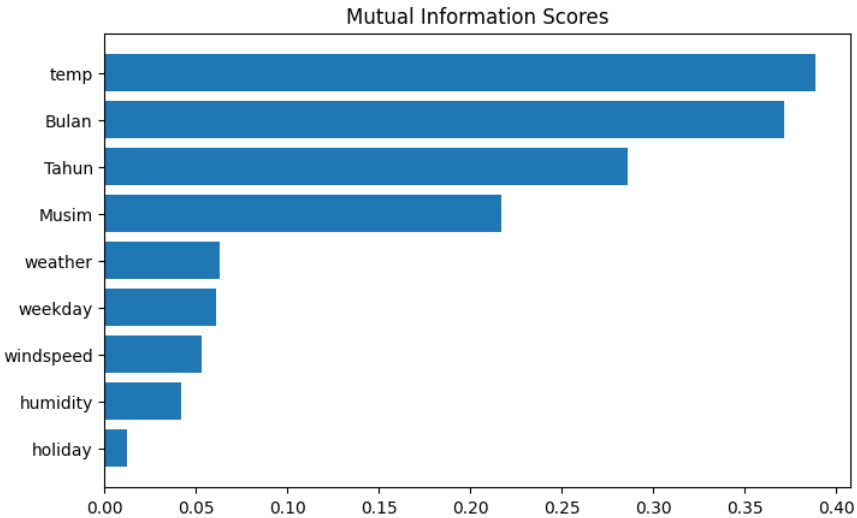
- Pada cuaca hujan ringan atau salju hubungannya adalah berbanding lurus namun karena datanya yang sedikit dan ralatnya yang besar maka hubungan tersebut masih sangat diragukan.

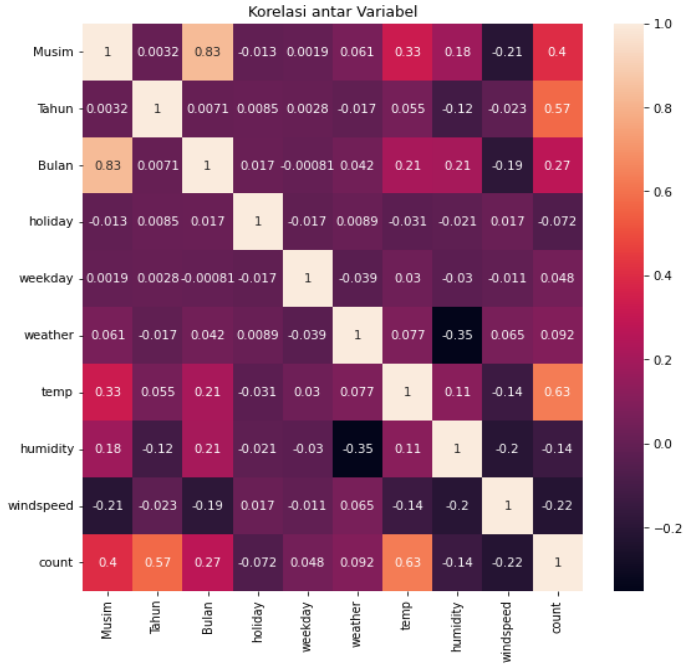
➢ **Apakah jenis musim mempengaruhi jumlah sepeda yang disewa?**



- Musim ternyata mempengaruhi jumlah sepeda yang disewa.
- Pada Musim Panas jumlah sepeda yang disewa paling banyak, sedangkan pada musim dingin adalah yang paling sedikit.

➢ **Faktor apa yang paling berpengaruh terhadap jumlah sepeda yang disewa?**





❖ Dari 2 parameter diatas yang saya gunakan, ternyata suhu menempati urutan pertama faktor paling berpengaruh terhadap jumlah sepeda yang disewa

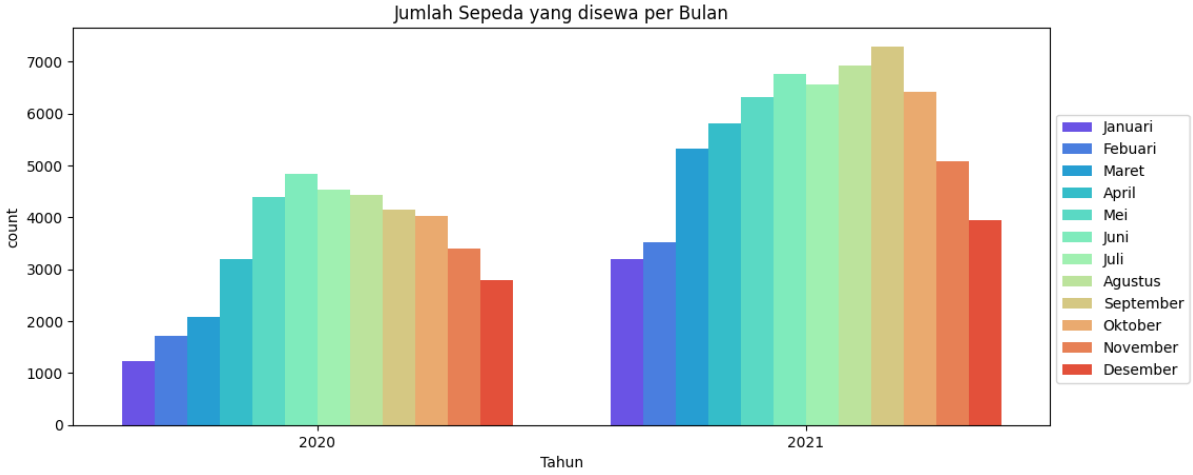
➢ **Kapan waktu yang tepat untuk perusahaan melakukan peningkatan layanan?**

❖ Dari grafik hubungan Hubungan antara jumlah sepeda yang disewa oleh pengguna terdaftar yang menyewa sepeda terhadap total penyewaan sepeda pada hari tertentu

maka perusahaan dapat mempertimbangkan untuk meningkatkan layanan pada hari selasa dan rabu.

- ❖ Dari hubungan jenis musim mempengaruhi jumlah sepeda yang disewa yang sudah dibahas di atas maka perusahaan dapat mempertimbangkan untuk meningkatkan layanan pada musim panas.

➢ **Apakah penyewaan sepeda tahun 2020 sama dengan tahun 2021? Bagaimana tren prediksi jumlah sepeda yang disewa pada 2 tahun kedepan?**



- ❖ Dari gambar histogram di atas dapat disimpulkan bahwa penyewaan sepeda tahun 2020 tidak sama dengan tahun 2021
- ❖ Tren penjualan yang ditemukan adalah tren naik dan seasonal yaitu mempunyai tren yang naik tiap tahun dan berubah naik turun secara seasonal selama setahun

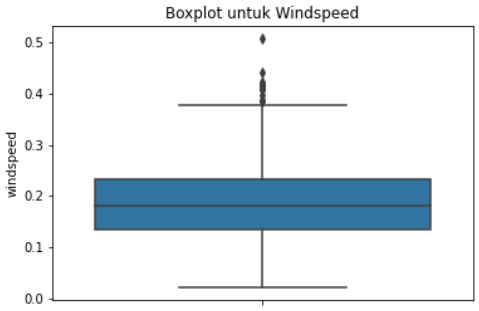
- Summary data

	Musim	Tahun	Bulan	holiday	weekday	weather	temp	humidity	windspeed	registered	count
count	731	731	731	731	731	731	731.000000	731.000000	731.000000	731.000000	731.000000
unique	4	2	12	2	7	3	NaN	NaN	NaN	NaN	NaN
top	3	2021	Januari	hari biasa	Wednesday	cerah berawan	NaN	NaN	NaN	NaN	NaN
freq	188	366	62	710	105	463	NaN	NaN	NaN	NaN	NaN
first	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	0.495385	0.627894	0.190486	3656.172367	4504.348837
std	NaN	NaN	NaN	NaN	NaN	NaN	0.183051	0.142429	0.077498	1560.256377	1937.211452
min	NaN	NaN	NaN	NaN	NaN	NaN	0.059130	0.000000	0.022392	20.000000	22.000000
25%	NaN	NaN	NaN	NaN	NaN	NaN	0.337083	0.520000	0.134950	2497.000000	3152.000000
50%	NaN	NaN	NaN	NaN	NaN	NaN	0.498333	0.626667	0.180975	3662.000000	4548.000000
75%	NaN	NaN	NaN	NaN	NaN	NaN	0.655417	0.730209	0.233214	4776.500000	5956.000000
max	NaN	NaN	NaN	NaN	NaN	NaN	0.861667	0.972500	0.507463	6946.000000	8714.000000

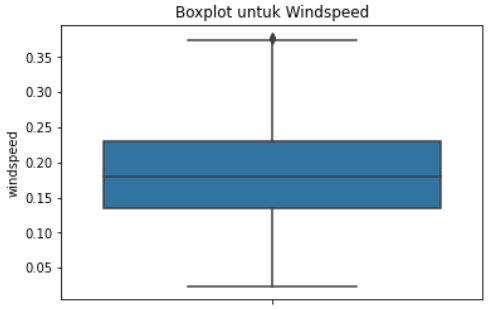
III. PEMODELAN AWAL :

MULTIPLE LINEAR REGRESSION

- Hal Pertama yang saya lakukan adalah melakukan cleaning data dengan membuang data-data pencilan.
  Seperti di bawah ini :



Data sebelum cleaning



Data Setelah Cleaning

- Setelah itu saya mendrop features *Tanggal* karena feature tersebut semuanya unique. Saya juga mendrop feature *Registered* karena saya mengansumsikan feature tersebut merupakan bagian dari data *Count*
- Kemudian saya mentransformasi feature-feature categorical dari string ke numerik.

➢ Dalam transformasi saya bagi menjadi dua metode transformasi
  - Pertama adalah Dengan Ordinal Encoding. Dalam Ordinal Encoding ini saya mengubah feature-feature categorical dari string menjadi numerik yang bertingkat misalnya 1,2,3,4,5,6 dst.
  - Kedua dengan OneHot Encoding yaitu metode yang merepresentasikan data bertipe kategori sebagai vektor biner yang bernilai integer, 0 dan 1, dimana semua elemen akan bernilai 0 kecuali satu elemen yang bernilai 1, yaitu elemen yang memiliki nilai kategori tersebut.
    - Dari setiap feature akan dipecah menjadi kolom-kolom data baru dari data-data yang unique pada setiap feature.
    - Satu kolom dari setiap feature akan saya drop.

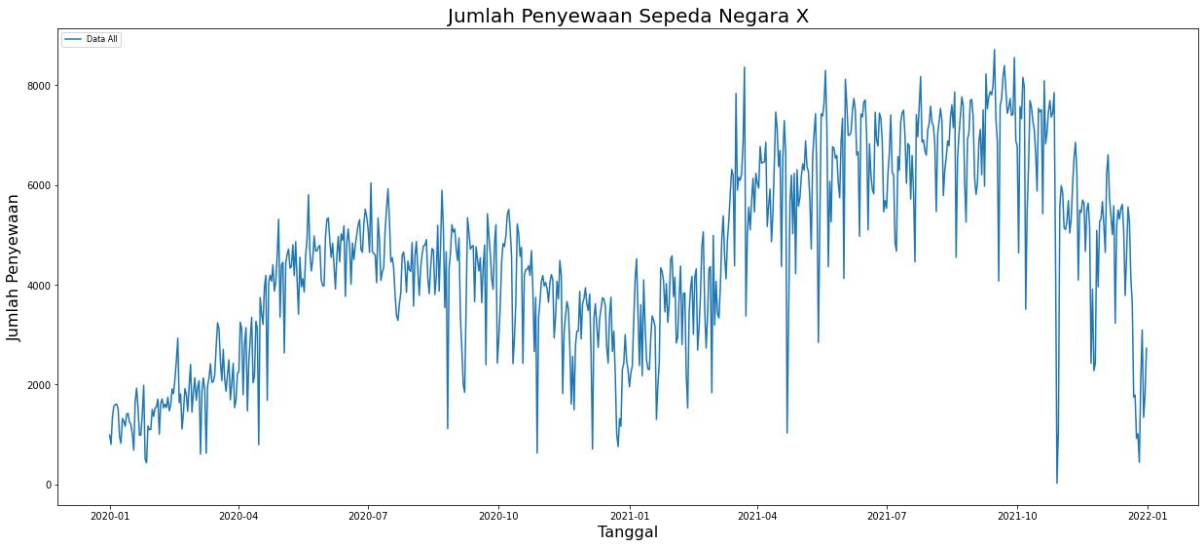
- Setelah itu saya membuat pemodelan dengan linear regresi berganda menggunakan bagian train pada masing-masing data transformasi dengan data train 80% dan data test 20%.
  - Hasil evaluasi untuk model regresi dengan transformasi encoding Ordinal :
    ```
    RMSE_Ordinal =  926.6365052307975
    R^2_Ordinal =  0.7483841334647081
    ```
  - Hasil evaluasi untuk model regresi dengan transformasi encoding One-Hot :
    ```
    RMSE_OneHot = 773.2093834685986
    R^2_OneHot = 0.8370084885149923
    ```
- Ternyata transformasi features categorical dari string ke numerik untuk metode encoding One-Hot lebih akurat dibanding metode encoding Ordinal.

SEASONAL AUTO REGRESSION INTEGRATED MOVING AVERAGE (SARIMA)

➢ Kita bisa melakukan pemodelan lagi namun kali dengan hanya menggunakan feature waktu dan variable Target, Dalam hal ini jumlah penyewaan sepeda (*Count*).

- Data yang terdiri dua kolom yang salah satu kolomnya adalah kolom waktu disebut dengan data Time Series. Seringkali kolom waktu ini menjadi indeks dalam data tersebut.
- Dalam melakukan analisis time series, kita melihat hubungan suatu data pada waktu tertentu terhadap jenis data yang sama namun pada waktu lainnya misalnya data hari ini dilihat hubungannya dengan data kemarin.
- Saya akan memprediksi jumlah penyewaan sepeda (*count*) untuk 2 tahun kedepan. Dalam hal ini saya akan mengambil kolom *Tanggal* dan *Count* dan kolom *tanggal* saya jadikan index.

Data penyewaan sepeda Dalam periode waktunya adalah :

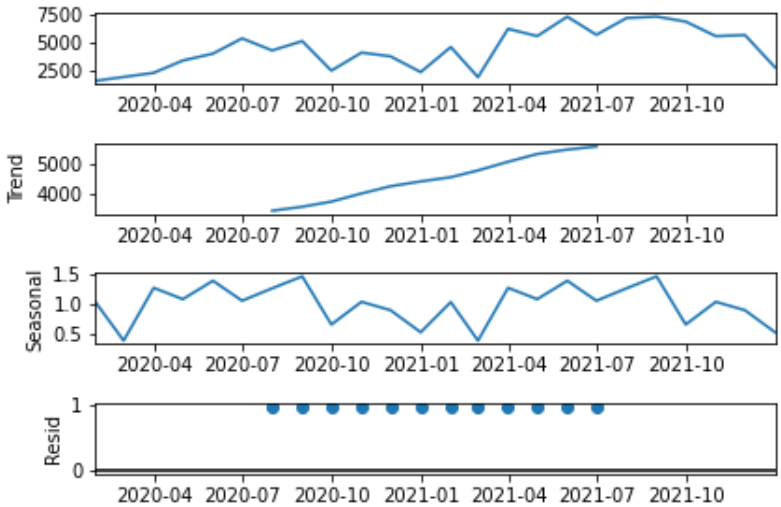


Kemudian kita cek apakah datanya stasioner dengan Augmented Dickey–Fuller test (ADF Test) :

```
1. ADF : -0.6731104071376196
```

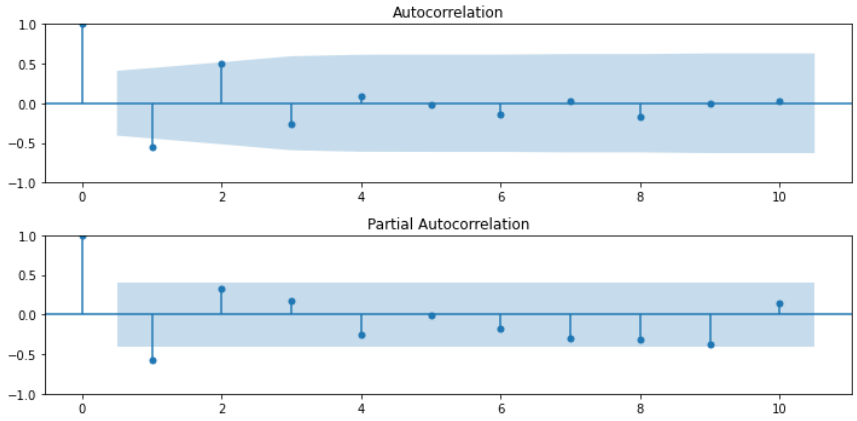
```
2. P-Value : 0.8535921588199734
```

Kemudian kita cek apakah mempunyai trend atau seasonalitas :



- Ternyata datanya tidak stasioner karena P-Value >> 0.05 untuk tes ADF dan mempunyai tren setiap tahun dan seasonal.

- Sehingga kita perlu membuat dataframe baru dengan cara membuat selisih tiap data terhadap data sebelumnya. Kemudian kita tes stasionernya dengan tes ADF:
  `1. ADF : -7.107513389843583`
  `2. P-Value : 4.017796208050733e-10`
  - P-Value sudah lebih kecil dari 0.05
- Sehingga kita bisa mengecek autocorrelation(ACF) dan partial autocorrelation(PACF):

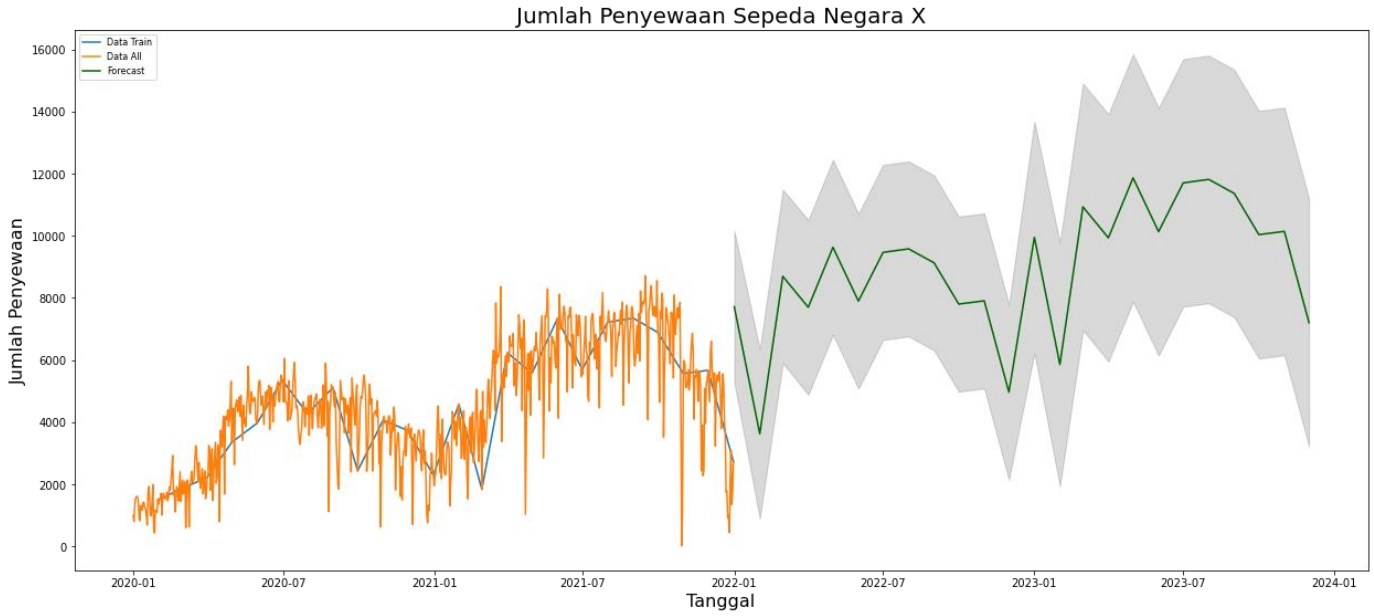


Dari ACF dan PACF tersebut dapat ditunjukkan bahwa lag 1 untuk PACF dan ACF signifikan dalam korelasinya.

- Sekarang masukklah kita ke tahap pemodelan. Karena datanya mempunyai tren naik setiap tahun dan mempunyai seasonality tiap 12 bulan, maka kita bisa memodelkan dan memprediksi jumlah penyewaan sepeda dengan metode SARIMA.
- Dalam memilih parameter, kita bisa memanfaatkan suatu algoritma untuk mencari parameternya secara otomatis dengan mencari nilai AIC terkecil, sehingga kita bisa dapatkan parameter terbaiknya adalah:
  `Best model: ARIMA(1,0,0)(0,1,0)[12] intercept`

  AIC 210.604

- Prediksi jumlah penyewaan sepeda untuk 2 tahun kedepan adalah :



- Dalam memasukkan ke pemodelan ime Series dan pengecekan stasionarity saya hanya mengambil satu data setiap bulan sampai data terakhirnya.

IV. Inferensi Hasil

Dengan menggunakan library statsmodels, didapat nilai – nilai sebagai berikut :

OLS Regression Results

Dep. Variable:	count	R-squared:	0.780
Model:	OLS	Adj. R-squared:	0.777
Method:	Least Squares	F-statistic:	278.7
Date:	Thu, 02 Jun 2022	Prob (F-statistic):	8.80e-226
Time:	11:12:28	Log-Likelihood:	-5899.5
No. Observations:	717	AIC:	1.182e+04
Df Residuals:	707	BIC:	1.186e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.039e+06	1.4e+05	-28.931	0.000	-4.31e+06	-3.76e+06
Musim	502.6312	57.276	8.776	0.000	390.180	615.083
Tahun	2000.4656	69.079	28.959	0.000	1864.841	2136.090
Bulan	-33.0677	17.904	-1.847	0.065	-68.219	2.083
holiday	-653.8899	202.507	-3.229	0.001	-1051.478	-256.302
weekday	20.6550	17.093	1.208	0.227	-12.904	54.214
weather	-62.5091	71.280	-0.877	0.381	-202.455	77.436
temp	5488.0917	200.345	27.393	0.000	5094.749	5881.435
humidity	-2855.9230	277.435	-10.294	0.000	-3400.618	-2311.228
windspeed	-3302.2055	494.911	-6.672	0.000	-4273.876	-2330.535

Omnibus:	75.049	Durbin-Watson:	1.005
Prob(Omnibus):	0.000	Jarque-Bera (JB):	138.192
Skew:	-0.660	Prob(JB):	9.82e-31
Kurtosis:	4.698	Cond. No.	8.28e+06

Inferensi Hasil untuk regresi linear berganda

Best model: ARIMA(1,0,0)(0,1,0)[12] intercept
Total fit time: 2.433 seconds

SARIMAX Results

Dep. Variable:	y	No. Observations:	24
Model:	SARIMAX(1, 0, 0)x(0, 1, 0, 12)	Log Likelihood	-102.302
Date:	Thu, 02 Jun 2022	AIC	210.604
Time:	12:43:57	BIC	212.059
Sample:	0	HQIC	210.066
	- 24		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	3352.2475	1303.741	2.571	0.010	796.962	5907.534
ar.L1	-0.4989	0.509	-0.980	0.327	-1.497	0.499
sigma2	1.553e+06	9.04e+05	1.718	0.086	-2.19e+05	3.33e+06

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	0.40
Prob(Q):	0.95	Prob(JB):	0.82
Heteroskedasticity (H):	1.82	Skew:	-0.18
Prob(H) (two-sided):	0.57	Kurtosis:	2.18

Inferensi Hasil untuk SARIMA

V. Lampiran

```
# Library yang saya perlukan nantinya
import seaborn as sns
import statsmodels.api as sm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from datetime import timedelta
%matplotlib inline
import seaborn as sns
from statsmodels.tsa.stattools import acf,pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from time import time
```

```
# Baca data dan cek spesifikasinya
df = pd.read_csv('Quiz3_Dataset.csv',parse_dates=['Tanggal'])
df.info()
df.nunique()
df_str = df
df_str[['Musim','Tahun']] = df_str[['Musim','Tahun']].apply(lambda x:
x.astype('str'))
df_str.describe(include='all').drop('Tanggal',axis=1)
```

```
# Summary of the data
df_str = df
df_str[['Musim','Tahun']] = df_str[['Musim','Tahun']].apply(lambda x:
x.astype('str'))
df_str.describe(include='all').drop('Tanggal',axis=1)
```

```
#Hubungan antara pengguna terdaftar yang menyewa sepeda terhadap total
penyewaan sepeda pada hari tertentu
sns.lmplot(data=df,x='registered',y='count',hue='weekday')
plt.title("Hubungan antara pengguna terdaftar yang menyewa sepeda terhadap
total penyewaan sepeda pada hari tertentu  ")
```

```
#Hubungan antara suhu dan jumlah sepeda yang disewa
sns.lmplot(data=df,x='temp',y='count')
plt.title('Hubungan antara suhu dan jumlah sepeda yang disewa')
```

```
#Hubungan antara suhu dan jumlah sepeda yang disewa dalam dua tahun
sns.lmplot(data=df,x='temp',y='count',hue='Tahun')
plt.title('Hubungan antara suhu dan jumlah sepeda yang disewa dalam dua
tahun')
```

```
#Hubungan antara kelembaban dan jumlah sepeda yang disewa
sns.lmplot(data=df,x='humidity',y='count')
plt.title('Hubungan antara kelembaban dan jumlah sepeda yang disewa')
```

```
### Hubungan antara kelembaban dan jumlah sepeda yang disewa jika cuaca
berbeda
sns.lmplot(data=df,x='humidity',y='count',col='weather')
```

```
# Hubungan antara jenis musim dengan jumlah sepeda yang disewa
musim = {'Musim':{1:'Dingin',2:'Semi',3:'Panas',4:'Gugur'}}
df_2= df.replace(musim)
```

```
sns.barplot(data=df_2,x='Musim',y='count')
plt.title('Hubungan antara jenis musim dengan jumlah sepeda yang disewa')
```

```
#Hubungan antara Kecepatan Angin dengan jumlah sepeda yang disewa
sns.lmplot(data=df,x='windspeed',y='count',)
plt.title('Hubungan antara Kecepatan Angin dengan jumlah sepeda yang disewa')
plt.figure(figsize=(12,5),dpi=100)
```

```
#Jumlah Sepeda yang disewa per Bulan
plt.figure(figsize=(12,5),dpi=100)
sns.barplot(x='Tahun',y='count',hue='Bulan',palette='rainbow',data=df,ci=None)
plt.legend(bbox_to_anchor =(1.15,0.8))
plt.title('Jumlah Sepeda yang disewa per Bulan')
```

```
#Jumlah Sepeda yang disewa per Musim
plt.figure(figsize=(12,5),dpi=100)
sns.barplot(x='Tahun',y='count',hue='Musim',palette='rainbow',data=df,ci=None)
plt.legend(bbox_to_anchor =(1.15,0.8))
plt.title('Jumlah Sepeda yang disewa per Musim')
```

```
#Persebaran Data
#Histogram untuk Temperatur  Terhadap Count
sns.histplot(data=df, x="temp", kde=True)
plt.title('Histogram untuk Temperatur  Terhadap Count')
```

```
#Boxplot untuk temperatur
sns.boxplot(data=df, y="temp")
plt.title('Boxplot untuk temperatur')
```

```
#Histogram untuk Humidity  Terhadap Count
sns.histplot(data=df, x="humidity", kde=True)
plt.title('Histogram untuk Humidity  Terhadap Count')
```

```
sns.boxplot(data=df, y="humidity")
plt.title('Boxplot untuk Humidity')
```

```
#Histogram untuk Windspeed  Terhadap Count
sns.histplot(data=df, x="windspeed", kde=True)
plt.title('Histogram untuk Windspeed  Terhadap Count')
```

```
#Boxplot untuk Windspeed
sns.boxplot(data=df, y="windspeed")
plt.title('Boxplot untuk Windspeed')
```

```
#Membersihkan Data Pencilannya :
var_outlier = ['humidity', 'windspeed']
for var in var_outlier:
 q1 = np.quantile(df[var], 0.25)
 q3 = np.quantile(df[var], 0.75)
 iqr = q3-q1
 upper_bound = q3+(1.5*iqr)
 lower_bound = q1-(1.5*iqr)
 df_clean = df[(df[var] > lower_bound) & (df[var] < upper_bound)]
```

```
#Boxplot untuk Windspeed setelah cleaning
sns.boxplot(data=df_clean, y="windspeed")
plt.title('Boxplot untuk Windspeed')
```

```
# Mengubah feature-feature kategorical dari string menjadi ordinal
X = df_clean.copy()
y = X.pop("count")
for colname in X.select_dtypes("object"):
    X[colname], _ = X[colname].factorize()

# All discrete features should now have integer dtypes (double-check this
before using MI!)
X = X.drop(['Tanggal','registered'],axis=1)
```

```
# Melihat seberapa signifikan suatu feature terhadap target dengan menggunakan
mutual information
from sklearn.feature_selection import mutual_info_regression

def make_mi_scores(X, y, discrete_features):
    mi_scores = mutual_info_regression(X, y,
discrete_features=discrete_features)
    mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)
    return mi_scores

discrete_features = X.dtypes == int
mi_scores = make_mi_scores(X, y, discrete_features)
def plot_mi_scores(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")


plt.figure(dpi=100, figsize=(8, 5))
```

```
plot_mi_scores(mi_scores) # show a few features with their MI scores
```

```
# Melihat seberapa signifikan suatu feature terhadap target dengan menggunakan
heatmap correlation
df_ordinal = pd.concat([X,y],axis=1)
plt.figure(figsize=(9,9))
sns.heatmap(df_ordinal.corr(), annot=True)
plt.title('Korelasi antar Variabel')
```

```
#Pemodelan jika menggunakan kategorical yang diubah menjadi ordinal
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=666)
lingress_model = LinearRegression()
lingress_model.fit(X_train,y_train)
```

```
#Evaluasi pemodelan jika menggunakan kategorical yang diubah menjadi ordinal
from sklearn.metrics import mean_squared_error,r2_score
test_prediction = lingress_model.predict(X_test)
MSE_ordinal = mean_squared_error(y_test,test_prediction)
RMSE_ordinal = np.sqrt(MSE_ordinal)
rsquared_score_ordinal = r2_score(y_test, test_prediction)
print('RMSE_Ordinal = ',RMSE_ordinal)
print('R^2_Ordinal = ',rsquared_score_ordinal)
```

```
#Summarry dari model jika menggunakan kategorical yang diubah menjadi ordinal
x = sm.add_constant(X)
model = sm.OLS(y, x).fit()
print(model.summary())
```

```
# Pemodelan jika menggunakan kategorical yang diubah menjadi ONE-HOT numeric
from sklearn.preprocessing import OneHotEncoder
df_onehot = df.copy()
df_onehot = pd.get_dummies(df_onehot,
columns=['Tahun','Bulan','holiday','weekday','weather','Musim'])
df_onehot = df_onehot.drop(['Tahun_2020','Bulan_Januari','holiday_hari
biasa','weekday_Monday','weather_berkabut','Musim_1'],axis=1)
df_onehot
```

```
X = df_onehot.copy()
y = X.pop("count")
X = X.drop(['Tanggal','registered'],axis=1)
```

```
#Evaluasi pemodelan jika menggunakan kategorical yang diubah menjadi ONE_HOT
numeric
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=666)
lingress_model = LinearRegression()
lingress_model.fit(X_train,y_train)
```

```
x = sm.add_constant(X)
model = sm.OLS(y, x).fit()
print(model.summary())
```

```
# Membuat indexer untuk mengambil satu data pada setiap bulan sampai data
terakhir
indexer = pd.date_range(start='28/01/2020', periods=24,
freq='M').to_frame().reset_index()
indexer['index'] = indexer[0].dt.date
indexer  = indexer.set_index('index')
indexer['month'] =indexer[0].dt.month
indexer = indexer.drop(0,axis=1)
```

```
#Mengambil data TimeSeries untuk pemodelan

#data per hari
df_count_all = df[['Tanggal','count']]
df_count_all.index = pd.to_datetime(df_count_all['Tanggal'])
df_count_all.drop(columns='Tanggal',inplace=True)
df_count = df[['Tanggal','count']]
df_count.index = pd.to_datetime(df_count['Tanggal'])
df_count.drop(columns='Tanggal',inplace=True)

##data per bulan
df_count = df_count.loc[indexer.index, :]
```

```
# Decompose seasonality and trend of the dataset
from statsmodels.tsa.seasonal import seasonal_decompose
decompose_data = seasonal_decompose(df_count,
model="multiplicative",period=12)

decompose_data.plot()
plt.figure(figsize=(21,9));
```

```
# Check if the dataset stationary:
from statsmodels.tsa.stattools import adfuller
```

```
dftest = adfuller(df_count, autolag = 'AIC')
print("1. ADF : ",dftest[0])
print("2. P-Value : ", dftest[1])
print("3. Num Of Lags : ", dftest[2])
print("4. Num Of Observations Used For ADF Regression and Critical Values
Calculation :", dftest[3])
print("5. Critical Values :")
for key, val in dftest[4].items():
    print("\t",key, ": ", val)
```

```
# Remove The Tren
first_diff = df_count.diff()[1:]
plt.figure(figsize=(21,9))
sns.lineplot(data=first_diff)
plt.title('Jumlah Penyewaan Sepeda Negara X',fontsize=20)
plt.ylabel('Jumlah Penyewaan',fontsize=16)
```

```
#Check the stationary after differeciancing
from statsmodels.tsa.stattools import adfuller
dftest = adfuller(first_diff, autolag = 'AIC')
print("1. ADF : ",dftest[0])
print("2. P-Value : ", dftest[1])
print("3. Num Of Lags : ", dftest[2])
print("4. Num Of Observations Used For ADF Regression and Critical Values
Calculation :", dftest[3])
print("5. Critical Values :")
for key, val in dftest[4].items():
    print("\t",key, ": ", val)
```

```
#Show ACF and PACF
import statsmodels.api as sm
fig,ax= plt.subplots(2,1, figsize=(10,5))
fig=sm.tsa.graphics.plot_acf(first_diff, lags=10, ax=ax[0])
fig=sm.tsa.graphics.plot_pacf(first_diff, lags=10, ax=ax[1])
plt.tight_layout()
plt.show()
```

```
# Mencari paramater terbaik untuk model SARIMA dengan mencari AIC terkecil

import pmdarima as pm

# Seasonal - fit stepwise auto-ARIMA
smodel = pm.auto_arima(df_count, start_p=1, start_q=1,
                       test='adf',
                       max_p=3, max_q=3, m=12,
                       start_P=0, seasonal=True,
```

```
                           d=None, D=1, trace=True,
                           error_action='ignore',
                           suppress_warnings=True,
                           stepwise=True)

smodel.summary()
```

```
#Memprediksi Jumlah penyewaan sepeda 2 tahun ke depan:

# Forecast
predictions = df_count.copy()
n_periods = 24
fitted, confint = smodel.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = pd.date_range(predictions.index[-1], periods = n_periods,
freq='MS')

# make series for plotting purpose
fitted_series = pd.Series(fitted, index=index_of_fc)
lower_series = pd.Series(confint[:, 0], index=index_of_fc)
upper_series = pd.Series(confint[:, 1], index=index_of_fc)

# Plot
plt.figure(figsize=(21,9))
plt.plot(predictions, label='Data Train')
plt.plot(df_count_all, label='Data All')
plt.plot(fitted_series, color='darkgreen',label='Forecast')
plt.fill_between(lower_series.index,
                 lower_series,
                 upper_series,
                 color='k', alpha=.15)

plt.title("SARIMA - Final Forecast of Drug Sales - Time Series Dataset")
plt.title('Jumlah Penyewaan Sepeda Negara X',fontsize=20)
plt.xlabel('Tanggal',fontsize=16)
plt.ylabel('Jumlah Penyewaan',fontsize=16)
plt.legend(loc='upper left', fontsize=8)
plt.show()
```