1. **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.

1. **Explanatory Data Analysis (EDA)**

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

Shape, arrow

Description automatically generated

* 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 :

Chart, scatter chart

Description automatically generatedScatter chart

Description automatically generated

* 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.
* Chart, scatter chart

  Description automatically generated **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?**

Scatter chart

Description automatically generated

* + 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?**

Chart, bar chart

Description automatically generated

* + Graphical user interface, application

    Description automatically generatedMusim ternyata mempengaruhi jumlah sepeda yang disewa.
  + Pada Musim Panas jumlah sepeda yang disewa paling banyak, sedangkan pada musim dingin adalah yang paling sedikit.
* A picture containing graphical user interface

  Description automatically generated**Faktor apa yang paling berpengaruh terhadap jumlah sepeda yang disewa?**

Chart, bar chart

Description automatically generated

* + 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?**

Chart

Description automatically generated

* + 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

Graphical user interface, text

Description automatically generated

1. **PEMODELAN AWAL :**

**MULTIPLE LINEAR REGRESSION**

* Hal Pertama yang saya lakukan adalah melakukan cleaning data dengan membuang data-data pencilan.

Seperti di bawah ini :

Chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

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 :**

Chart, line chart

Description automatically generated

**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 :**

Background pattern

Description automatically generated

* 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):

A picture containing timeline

Description automatically generated

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 :

Chart

Description automatically generated

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

1. **Inferensi Hasil**

**A screenshot of a computer

Description automatically generated with low confidence**Dengan menggunakan library statsmodels, didapat nilai – nilai sebagai berikut :

**Graphical user interface

Description automatically generated**

Inferensi Hasil untuk regresi linear berganda Inferensi Hasil untuk SARIMA

1. **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()