Prediksi Permintaan Sewa Sepeda di London Pada Hari Libur/Akhir Pekan Menggunakan Algoritma Naive Bayes dan Random Forest

Aljevan Komala (00000044016)[1], Jerrell Susilo (00000044016)[2], Thomas Januardy (00000046001)[3]
Information System
Universitas Multimedia Nusantara
Tangerang, Indonesia

***Abstract*—Data merupakan salah satu hal yang sangat penting di era segala teknologi seperti saat ini untuk membantu kelangsungan suatu proses. Salah satu proses yang menggunakan data dalam kegiatan sehari-hari adalah sistem *bike-sharing* dengan menggunakan sistem rental. Permintaan *bike-sharing* yang terus meningkat di kota London, maka dibutuhkan suatu prediksi seberapa besar kejadian pada permintaan *bike sharing* di masa depan. Karena diketahui bahwa penggunaan sepeda bisa menjadi bisa menjadi salah satu solusi penghematan energi, sehingga perlu adanya peningkatan sistem *bike-sharing* di kota London. Dari data yang terkumpul, tujuan dari penelitian ini adalah untuk memprediksi seberapa besar kejadian *bike-sharing* di masa depan berdasarkan penggunaan dan permintaan pada akhir pekan atau hari libur. Model prediksi yang digunakan dalam penelitian ini menggunakan dua algoritma klasifikasi, yaitu Naive Bayes dan Random Forest. Variabel target yang digunakan sebagai acuan dicantumkan pada kolom "is_holiday" (hari libur) dan "is_weekend" (akhir pekan) yang menggunakan variabel dependen lainnya untuk menentukan prediktor di masa mendatang. Variabel yang digunakan tersebut mendukung dalam menebak pola dan melihat frekuensi penggunaan sepeda yang paling dominan, khususnya pada hari libur maupun di akhir pekan dalam mengetahui penggunaan sepeda yang paling sering dan sering terjadi.**

***Index Terms*—london, bike, bike sharing, cycling, classification, prediction, naive bayes, random forest, CRISP-DM**

I. INTRODUCTION

Saat ini, *data mining* merupakan suatu hal yang sangat dibutuhkan dalam memperoleh informasi dari sebuah data dengan cara menggunakan suatu metode untuk untuk memperoleh pola data dan kemudian dianalisis[1]. Menurut ahli (Suntoro, 2019), *data mining* adalah proses untuk mendapatkan informasi yang berguna dari basis data yang besar dan perlu diekstraksi agar menjadi informasi baru dan dapat membantu dalam pengambilan keputusan[2]. Dalam *data mining*, klasifikasi merupakan salah satu teknik dalam membangun model dari sampel data yang dimiliki. Klasifikasi umumnya menggunakan beberapa algoritma diantaranya seperti Naïve Bayes, KNN, Decision Tree, Support Vector Machine, dan masih banyak lagi.

Bike-sharing dapat diartikan sebagai suatu sistem yang menyediakan layanan penyewaan sepeda kepada masyarakat. Masyarakat yang ingin menyewa sepeda dapat melakukan transaksi pada tempat sepeda yang biasanya tersebar di area publik di dalam kota[3].

Dalam hal ini, terdapat kegiatan *bike-sharing* berupa persewaan sepeda di kota London. Sepeda merupakan salah satu alat transportasi yang paling umum dan sering digunakan oleh masyarakat di kota London. Oleh karena itu, tidak heran jika banyak peluang usaha dan bisnis berupa persewaan dengan sistem berbagi kendaraan (sepeda) kepada konsumen yang membutuhkannya di kota London. Seperti yang diketahui bahwa benua Eropa tempat kota London berada memiliki musim yang lebih beragam, seperti musim dingin, musim semi, musim panas, dan musim gugur. Mobilitas dan aktivitas masyarakat juga terpengaruh dan selalu beradaptasi seiring berjalannya musim saat ini, terutama penggunaan sepeda sebagai alat transportasi sehari-hari. Dari data yang terkumpul, tujuan dari penelitian ini adalah untuk memprediksi seberapa besar kejadian *bike-sharing* di waktu yang akan datang berdasarkan penggunaan dan permintaan pada akhir pekan dan hari libur. Prediksi penggunaan kendaraan bersepeda bisa menjadi salah satu solusi penghematan energi, di mana sepeda merupakan kendaraan yang tidak membutuhkan energi, tetapi manusia yang menjadi sumber energi penggeraknya sendiri.

Untuk itu penelitian yang ingin dilakukan adalah untuk memprediksi tingkat penggunaan sepeda di tempat persewaan sepeda yang terjadi di kota London berdasarkan acuan pada hari libur dan akhir pekan. Model prediksi yang digunakan dalam penelitian ini menggunakan dua algoritma klasifikasi, yaitu Naive Bayes dan Random Forest. Variabel acuan yang digunakan dicantumkan pada kolom "is_holiday" (hari libur) dan "is_weekend" (akhir pekan) untuk

menggunakan variabel dependen lainnya untuk menentukan prediktor di masa mendatang.

II. LITERATURE STUDY

A. Landasan Teori

Penelitian ini menggunakan dua metode algoritma yang digunakan dalam penelitian ini untuk memprediksi dalam penelitian kami. Naive Bayes dan Random Forest adalah dua macam algoritma yang digunakan. Dua algoritma yang digunakan dalam penelitian ini akan dijelaskan secara umum dalam bab ini, bersama dengan sejumlah referensi untuk penelitian lain yang menggunakan teknik yang sama.

Data Mining

Data Mining adalah langkah dalam proses penemuan pengetahuan secara keseluruhan yang dapat digambarkan sebagai proses penggalian atau penambangan pengetahuan dari sejumlah besar data[4]. *Data mining* disebut sebagai bentuk dari penemuan pengetahuan yang begitu penting karena dapat memecahkan masalah dalam domain tertentu menggunakan algoritma-algoritma tertentu. Salah satu contoh teknik *data mining* yang dapat digunakan dalam bidang pendidikan, yaitu untuk meningkatkan pemahaman tentang proses pembelajaran dalam mendalami fokus pada identifikasi, ekstraksi dan evaluasi variabel yang terkait dengan proses pembelajaran siswa.

CRISP-DM

CRISP-DM atau *Cross-Industry Standard Process for Data Mining* dapat dijabarkan sebagai suatu model proses *data mining* yang paling sering digunakan dalam pengembangan *data mining* itu sendiri. CRISP-DM sendiri pada awalnya dibangun pada tahun 1966 oleh 5 perusahaan yakni Integral Solution LTD (ISL), Teradata, Daimler AG, NCR Corporation, dan OHRA. Seiring dengan berjalannya waktu, *framework* ini dikembangkan oleh berbagai organisasi dan perusahaan di Eropa dan dijadikan sebagai standar dalam *data mining*. Fase-fase yang dimiliki CRISP-DM membuat pengolahan sebuah data menjadi lebih terdefinisi dengan jelas serta lebih efisien dalam penyelesaiannya. Proses *data mining* dalam CRISP-DM kerap digunakan untuk memecahkan masalah bisnis dan menjadi jawaban solutif dalam mengolah sekumpulan data menjadi informasi yang bernilai. Fase-fase yang dimiliki oleh CRISP-DM antara lain adalah *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, dan *deployment*. Masing-masing tahap tersebut memiliki fungsi dan tujuannya masing-masing, sampai pada akhirnya menghasilkan keputusan akhir.

Naïve Bayes

Naive Bayes adalah pengklasifikasi probabilistik sederhana yang menghitung satu set probabilitas dengan menjumlahkan frekuensi dan kombinasi nilai dari kumpulan data yang diberikan. Algoritma menggunakan Teorema Bayes dan mengasumsikan semua atribut independen atau noninterdependen diberikan oleh nilai variabel kelas .Definisi lain mengatakan Naive Bayes adalah pengklasifikasi dengan metode probabilitas dan statistik yang dibawa oleh ilmuwan Inggris Thomas Bayes, memprediksi peluang di masa depan berdasarkan pengalaman sebelumnya[5]. Rumus persamaan Naïve Bayes adalah sebagai berikut:

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)} (1)$$

Gambar 1. Rumus Probabilitas Naive Bayes

Keterangan :

a. X: Kriteria suatu kasus berdasarkan masukan
b. Ci: Kelas solusi pola ke-i, dimana i adalah jumlah label kelas
c. P(Ci|X): Probabilitas kemunculan label kelas Ci dengan kriteria masukan X
d. P(X|Ci): Probabilitas kriteria masukan X dengan label kelas Ci
e. P(Ci): Probabilitas label kelas Ci

Random Forest

Metode Random Forest merupakan metode yang dapat meningkatkan hasil akurasi, karena dalam membangun simpul anak untuk setiap node dilakukan secara acak[6]. Random Forest digunakan untuk membuat suatu *decision tree* atau pohon keputusan yang di dalamnya terdiri dari beberapa bagian, seperti *root node*, *internal node*, dan *leaf node* dengan mengambil atribut dan data secara acak sesuai dengan syarat atau ketentuan yang diterapkan pada implementasinya. *Root node* sendiri merupakan suatu simpul yang terletak pada posisi paling atas/puncak atau juga biasa disebut dengan akar dari *decision tree* atau pohon keputusan. *Leaf node* merupakan simpul terakhir yang hanya memiliki satu *input* dan tidak menghasilkan *output*. Berikut merupakan penulisan rumus dari metode Random Forest:

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

Gambar 2. Rumus Algoritma Random Forest

B. Penelitian Terdahulu

Dalam penelitian ini, digunakan beberapa jurnal terdahulu yang membahas sebuah masalah dan menggunakan algoritma yang sama dalam penyelesaian masalahnya. Penelitian terdahulu ini

sekaligus menjadi referensi dan panduan bagi kami dalam menyelesaikan masalah dalam penelitian prediksi menggunakan studi kasus *London Bike Sharing* ini. Dengan adanya penelitian terdahulu, maka penelitian yang kami lakukan ini dapat lebih terarah dengan cara melihat algoritma yang digunakan dalam bentuk pemecahan masalah yang ada, yang telah dilakukan dan selesai dalam penelitian terdahulu.

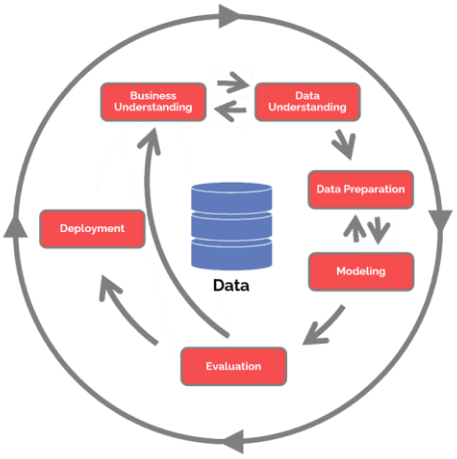
Adapun jurnal dari penelitian terdahulu yang kami gunakan sebagai acuan dan referensi karena kami menilai penelitian tersebut menggunakan algoritma yang sama dalam menyelesaikan masalahnya serta dari sumber data yang digunakan di mana dalam salah penelitian tersebut menggunakan algoritma Naive Bayes untuk mengklasifikasikan data nasabah asuransi. Peneliti ingin mengetahui lancar, kurang lancar, atau tidak lancarnya nasabah dalam membayar premi di asuransi tersebut menggunakan metode Naive Bayes sehingga mereka dapat mengetahui dan dapat mengambil keputusan untuk menerima atau menolak calon nasabah tersebut[7].

Pada metode algoritma Random Forest, memprediksi harga ponsel, di mana permasalahan pada penelitian tersebut, peneliti ingin melakukan klasifikasi untuk memprediksi harga ponsel dengan spesifikasi yang diberikan, dan setelah melakukan berbagai pengujian menggunakan metode klasifikasi Random Forest, didapatkan nilai akurasi sebesar 81% dan dinilai hasil tersebut lebih baik dibandingkan dengan percobaan yang telah dilakukan sebelumnya[8].

III. METHODOLOGY

Metode penelitian adalah langkah yang dilakukan dalam rangka untuk mengumpulkan informasi atau data, serta melakukan evaluasi pada data yang telah didapatkan tersebut. Metode penelitian memberikan suatu gambaran rancangan penelitian yang meliputi prosedur penelitian dan langkah-langkah yang harus dikerjakan, waktu penelitian, sumber data, dan dengan langkah apa data-data tersebut diperoleh dan tahap selanjutnya diolah dan dianalisis untuk mendapatkan suatu kesimpulan. Untuk menyesuaikan tujuan dan mendukung penelitian ini, metodologi utama menggunakan CRISP-DM sebagai standar umum dari penelitian di bidang *data science*.

CRISP-DM atau *Cross-Industry Standard Process for Data Mining* yang berdasar pada standar suatu konsep model yang digunakan dalam proses *data mining* yang membentuk gambaran siklus hidup dari *data mining* hingga ke penyelesaian permasalahan bisnis[9]. Rangkaian tahapan yang dimiliki oleh siklus CRISP-DM ini terdiri dari 6 macam, yakni *Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation*, dan *Deployment.*



Gambar 3. Siklus CRISP-DM

1. *Business Understanding*
   Business Understanding atau pemahaman pada bisnis yang ingin dimaksud, merupakan tahapan dalam menyiapkan strategi dan perancangan untuk menyelesaikan permasalahan yang dihadapi, seperti saat menentukan langkah-langkah yang dibutuhkan untuk menggapai suatu tujuan dan memenuhi keinginan dari segala persiapan yang telah disiapkan.

   Pada kasus bisnis penelitian ini, diperoleh dari perusahaan rental transportasi berupa *bike-sharing* di kota London. Potensi dalam meningkatnya/menurunnya penggunaan *bike-sharing* terkait dengan beberapa faktor, seperti penggunaan saat hari libur/*weekend*. Maka dari itu, tujuan dari penelitian ini adalah untuk memprediksi tingkat penggunaan sepeda di tempat *bike-sharing* di kota London berdasarkan acuan pada hari libur dan akhir pekan, dengan strategi perolehan data yang didapat dari situs Kaggle berisikan data-data dari setiap *record* penggunaan *bike-sharing*.

2. *Data Understanding*
   Data Understanding atau Pemahaman pada Data, merupakan suatu fase pengumpulan data secara kolektif untuk mendukung pemahaman model. Pengumpulan ini juga dilakukan dengan syarat kualifikasi data, dengan kualitas dan limitasi yang dimiliki oleh suatu data sehingga dapat diketahui kelebihan dan kekurangan nilai yang dimiliki oleh suatu data.

   Berdasarkan data yang diperoleh dari Kaggle, dapat diketahui jumlah variabel dan masing-masing penjabaran setiap data, mulai dari tipe, nama, dan setiap *record* datanya. Data *bike-sharing* yang digunakan terdiri atas 17.415 *record* data dengan 10 kolom, berjumlah 8 untuk tipe data *float*, 1 untuk

integer, dan 1 untuk *datetime.* Selain itu, dilakukan eksplorasi data seperti melihat perbandingan atau frekuensi dari masing-masing variabel menggunakan visualisasi plot.

3. *Data Preparation*
   Data Preparation atau Persiapan Data, yaitu tahap pengolahan data, dengan cara data yang sudah diambil akan dimanipulasi dan dikonversi sedemikian rupa sehingga membentuk bentuk (*form*) yang mudah dipahami dan dapat dilanjutkan ke proses selanjutnya. Konversi tersebut memiliki kerelatifan terhadap kebutuhan yang diinginkan untuk hasil analisa. Salah satu bentuk manipulasi dan konversi adalah dengan identifikasi *missing value*, jika terdapat *missing value* yang diakibatkan oleh *error* saat penginputan data, maka keseluruhan *value* tersebut akan dihapus.

   Persiapan data awal dengan menyeleksi variabel apa yang diperlukan pada kebutuhan penelitian, yakni "timestamp", "cnt", "t1", "t2", "hum", "wind_speed", "weather_code", "is_holiday", "is_weekend", dan "season". Pada variabel "timestamp", dilakukan 'pemecahan' pada struktur *datetime* ke dalam beberapa variabel yang spesifik untuk mendefinisikan waktu yang lebih rinci. Variabel-variabel yang telah terseleksi akan disesuaikan, mulai dari tipe hingga nilainya sedemikian rupa untuk kebutuhan model. Selain itu, untuk mencegah data yang keliru, dilakukan *data cleaning* untuk memastikan tidak adanya *missing value* maupun *null* pada data.

4. *Modeling*
   Modeling, yaitu tahap pembuatan model dengan cara penentuan teknik data mining yang sesuai untuk data yang telah dikumpulkan dan dievaluasi/diperbaiki sebelumnya. Teknik *data mining* terdapat berbagai macam yang dapat dilakukan, contohnya dengan menggunakan model klasifikasi atau prediksi sesuai karakteristik data dan output yang diinginkan dari penelitian analisis model.

   Penelitian ini menggunakan algoritma Naive Bayes dan Random Forest untuk pemodelannya. Algoritma yang dipilih merupakan algoritma dengan tujuan klasifikasi berbasis *supervised learning* untuk kebutuhan prediktif. Model menerapkan pembagian rasio data 70:30 pada *data training* dan *data testing*. Pemodelan menggunakan bahasa pemrograman dari Python yang memiliki fitur *library* untuk kebutuhan bidang *data science* berdasarkan aplikasi distribusi Jupyter Notebook.

5. *Evaluation*
   Evaluation, yaitu tahap evaluasi terhadap hasil dan kinerja dari proses *modeling* yang dilakukan sebelumnya. Proses penilaian yang dapat diberikan dapat berdasarkan kategori kualitatif maupun kuantitatif, bergantung pada tujuan. Tahap evaluasi ini tentu dapat memberikan bantuan dan dukungan, khususnya terhadap pemahaman tujuan bisnis dan segala macam kesesuaian metode rangkaian yang sudah dilakukan sebelumnya.

   Pembuatan pemodelan yang terdiri dari dua macam algoritma, diperlukan perbandingan antar model untuk mengetahui pengujian yang terbaik di antaranya. Pengujian tersebut dapat dilakukan dengan evaluasi model yang menggunakan metode *Confusion Matrix*, *Accuracy Metrics*, dan *Cross Validation* dari *library* Scikit-Learn. *Confusion Matrix* untuk melihat hasil akhir dan pengujian yang sesuai dengan penggunaan model dengan algoritma Naive Bayes dan Random Forest, serta pengukuran akurasi ketepatan dari sebuah model yang dapat diukur dengan pengujian berdasarkan *Accuracy Metrics.*

6. *Deployment*
   Deployment atau tahap di mana fase yang telah didapatkan dari fase sebelumnya diwujudkan dengan penggunaan secara *real* agar dapat membantu bisnis dengan tujuan yang telah dirumuskan. *Deployment* mengacu kembali pada titik awal proses, yaitu tentang apa yang diinginkan dan dipahamkan oleh suatu bisnis sehingga dapat memproses tujuannya. Dengan itu, terbentuk sebuah model dan proses untuk pengklasifikasian data sehingga dapat menciptakan sebuah informasi yang berguna untuk menjadi nilai tambahan sebagai faktor penentu dari suatu keputusan bisnis untuk menghasilkan "*return on investment"*. Namun sebaliknya, jika hasil yang didapat belum dapat memenuhi tujuan, rangkaian CRISP-DM ini dapat diulangi dari awal lagi.

IV. RESULT AND ANALYSIS

a. Business Understanding

Tahap pertama dengan melakukan pemahaman dan menemukan tujuan dari penelitian terhadap kasus bisnis. Penelitian ini berorientasi pada prediksi apakah adanya peningkatan jumlah *bike sharing* saat hari libur

dan *weekend* berdasarkan pengolahan data indikator yang ada.

Kebutuhan tentang pemahaman bisnis dapat diperoleh dengan menggunakan *dataset* yang tersedia pada Kaggle mengenai data *bike sharing* di kota London pada tahun 2015-2016 yang akan digunakan untuk membuat model prediksi.

b. Data Understanding

Tahap kedua, setelah penentuan tujuan dan pemahaman pada kebutuhan penelitian ini, eksplorasi dan pengecekan data diperlukan untuk pemahaman akan *dataset* yang digunakan. Data yang digunakan adalah data penggunaan *bike-sharing* di kota London dengan *record* sebanyak 17.415 terdiri atas 10 kolom, berjumlah 8 untuk tipe data *float*, 1 untuk *integer*, dan 1 untuk *datetime.*

Berikut adalah *preview dataset* yang digunakan:

	A	B	C	D	E	F	G	H	I	J
1	timestamp	cnt	t1	t2	hum	wind_spe	weather_	is_holiday	is_weeke	season
2	1/4/2015 0:0	182	3	2	93	6	3	0	1	3
3	1/4/2015 1:0	138	3	2.5	93	5	1	0	1	3
4	1/4/2015 2:0	134	2.5	2.5	96.5	0	1	0	1	3
5	1/4/2015 3:0	72	2	2	100	0	1	0	1	3
6	1/4/2015 4:0	47	2	0	93	6.5	1	0	1	3
7	1/4/2015 5:0	46	2	2	93	4	1	0	1	3
8	1/4/2015 6:0	51	1	-1	100	7	4	0	1	3
9	1/4/2015 7:0	75	1	-1	100	7	4	0	1	3
10	1/4/2015 8:0	131	1.5	-1	96.5	8	4	0	1	3
11	1/4/2015 9:0	301	2	-0.5	100	9	3	0	1	3
12	1/4/2015 10:0	528	3	-0.5	93	12	3	0	1	3
13	1/4/2015 11:0	727	2	-1.5	100	12	3	0	1	3
14	1/4/2015 12:0	862	2	-1.5	96.5	13	4	0	1	3
15	1/4/2015 13:0	916	3	-0.5	87	15	3	0	1	3
16	1/4/2015 14:0	1039	2.5	0	90	8	3	0	1	3
17	1/4/2015 15:0	869	2	-1.5	93	11	3	0	1	3
18	1/4/2015 16:0	737	3	0	93	12	3	0	1	3
19	1/4/2015 17:0	594	3	0	93	11	3	0	1	3
20	1/4/2015 18:0	522	3	1.5	93	6.5	3	0	1	3
21	1/4/2015 19:0	379	3	1	93	7	3	0	1	3
22	1/4/2015 20:0	328	3	3	93	4	3	0	1	3
23	1/4/2015 21:0	221	3	2.5	93	5	4	0	1	3

Gambar 4. *Dataset*

Berikut merupakan deskripsi pada setiap kolom variabel dalam *dataset* yang digunakan:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17414 entries, 0 to 17413
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   timestamp     17414 non-null  object
 1   cnt           17414 non-null  int64
 2   t1            17414 non-null  float64
 3   t2            17414 non-null  float64
 4   hum           17414 non-null  float64
 5   wind_speed    17414 non-null  float64
 6   weather_code  17414 non-null  float64
 7   is_holiday    17414 non-null  float64
 8   is_weekend    17414 non-null  float64
 9   season        17414 non-null  float64
dtypes: float64(8), int64(1), object(1)
memory usage: 1.3+ MB
```

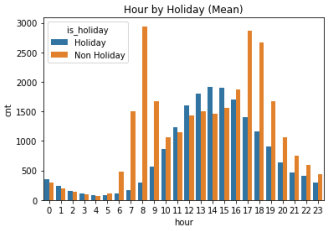
	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0
...	...	...	...	...	...	...	...	...	...	...
17409	2017-01-03 19:00:00	1042	5.0	1.0	81.0	19.0	3.0	0.0	0.0	3.0
17410	2017-01-03 20:00:00	541	5.0	1.0	81.0	21.0	4.0	0.0	0.0	3.0
17411	2017-01-03 21:00:00	337	5.5	1.5	78.5	24.0	4.0	0.0	0.0	3.0
17412	2017-01-03 22:00:00	224	5.5	1.5	76.0	23.0	4.0	0.0	0.0	3.0
17413	2017-01-03 23:00:00	139	5.0	1.0	76.0	22.0	2.0	0.0	0.0	3.0

17414 rows × 10 columns

Gambar 5. Deskripsi *Dataset*

Berikut beberapa visualisasi data yang dibuat sebagai gambaran terhadap data yang akan dimodelkan:

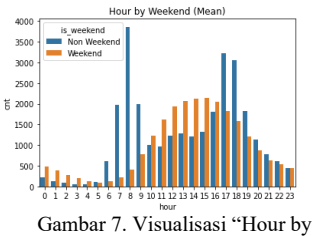
```
# hour by holiday (mean)
holiday_map = {0: 'Non Holiday', 1:'Holiday'}
london_viz['is_holiday'] = london_viz['is_holiday'].map(holiday_map)

london_holiday = london_viz.groupby(['hour', 'is_holiday'])['cnt'].mean().reset_index()
sns.barplot(data=london_holiday, x='hour', y='cnt', hue='is_holiday')
plt.title("Hour by Holiday (Mean)")
plt.show()
```



Gambar 6. Visualisasi "Hour by Holiday (Mean)"

Contoh visualisasi di atas adalah plot berbentuk *bar* dengan gambar pengukuran penggunaan *bike-sharing* per jamnya berdasarkan acuan hari libur. Dari plot tersebut terlihat bahwa pada hari bukan libur, kepadatan penggunaan terjadi pada saat di pagi hari mendekati rata-rata 3.000 penggunaan (saat jam berangkat kerja) dan saat sore-malam hari (saat jam pulang kerja) yang menandakan demografi masyarakat dalam menggunakan transportasi umum. Sebaliknya, saat hari libur, justru penggunaan meningkat saat di tengah siang hari dengan intensitas semakin ramai dan padat kegiatan di masyarakat.

```
# hour by weekend (mean)
weekend_map = {0: 'Non Weekend', 1:'Weekend'}
london_viz['is_weekend'] = london_viz['is_weekend'].map(weekend_map)

london_weekend = london_viz.groupby(['hour', 'is_weekend'])['cnt'].mean().reset_index()
sns.barplot(data=london_weekend, x='hour', y='cnt', hue='is_weekend')
plt.title("Hour by Weekend (Mean)")
plt.show()
```



Gambar 7. Visualisasi "Hour by Weekend (Mean)"

Contoh visualisasi di atas adalah plot berbentuk *bar* dengan gambar pengukuran penggunaan *bike-sharing* per jamnya berdasarkan acuan akhir pekan. Dari visualisasi sebelumnya terlihat mirip, yang dapat ditarik kesimpulan

bahwa rata-rata pada hari yang bukan di akhir pekan, penggunaan ramai pada saat jam pagi berangkat kerja hampir mencapai rata-rata 4.000 penggunaan dan jam sore-malam saat pulang kerja. Sebaliknya, pada hari akhir pekan, penggunaan *bike-sharing* cenderung baru meningkat saat di tengah siang hari.

c. Data Preparation

Tahap *Data Preparation* atau *Data Preprocessing* yang dibutuhkan agar kesesuaian data yang lebih terstruktur untuk pengembangan model yang lebih akurat. Persiapan yang pertama dilakukan adalah dengan mengubah tipe data pada kolom data agar mempermudah pengolahan data menjadi *integer*, antara lain "weather_code", "is_holiday", "is_weekend", dan "season" (untuk indikasi setiap metadata *code* yang sesuai). Kolom variabel "timestamp" juga disesuaikan menggunakan fungsi dari *pandas*, untuk mengubah format menjadi data menjadi *datetime*.

```
# data preprocessing

# convert dtypes
london = london.astype({'weather_code':'int64'})
london = london.astype({'is_holiday':'int64'})
london = london.astype({'is_weekend':'int64'})
london = london.astype({'season':'int64'})

london['timestamp'] = pd.to_datetime(london['timestamp'])

print(london.dtypes)
print(london.describe)
```

```
timestamp       datetime64[ns]
cnt                      int64
t1                     float64
t2                     float64
hum                    float64
wind_speed             float64
weather_code             int64
is_holiday               int64
is_weekend               int64
season                   int64
dtype: object
```

Gambar 8. *Data Preparation*

Untuk memaksimalkan hasil model dengan penggunaan pendukung variabel waktu, pada kolom variabel "timestamp" akan dibagi ke dalam beberapa kolom variabel yang memiliki masing-masing rincian waktunya, antara lain "year", "month", "day", "day_of_week", dan "hour" pada dimensi *dataframe* yang baru (duplikasi).

```
# duplicate dataframe
london_time = london.copy()

# specific timestamp dtype into columns
london_time['year'] = london_time['timestamp'].dt.year
london_time['month'] = london_time['timestamp'].dt.month
london_time['day'] = london_time['timestamp'].dt.day
london_time['day_of_week'] = london_time['timestamp'].dt.dayofweek
london_time['day_of_week'] += 1 #changing the index to map monday as 1 instead of 0
london_time['hour'] = london_time['timestamp'].dt.hour

london_df = ['year', 'month', 'season', 'day', 'day_of_week', 'is_weekend', 'is_holiday',
             'hour', 'cnt', 't1', 't2', 'weather_code', 'hum', 'wind_speed']
london_time = london_time[london_df]

london_time
```

	year	month	season	day	day_of_week	is_weekend	is_holiday	hour	cnt	t1	t2	weather_code	hum	wind_speed
0	2015	1	3	4	7	1	0	0	182	3.0	2.0	3	93.0	6.0
1	2015	1	3	4	7	1	0	1	138	3.0	2.5	1	93.0	5.0
2	2015	1	3	4	7	1	0	2	134	2.5	2.5	1	96.5	0.0
3	2015	1	3	4	7	1	0	3	72	2.0	2.0	1	100.0	0.0
4	2015	1	3	4	7	1	0	4	47	2.0	0.0	1	93.0	6.5

Gambar 9. *Data Preparation (timestamp)*

Pengecekan terhadap data dalam *data cleaning* juga diterapkan untuk mencari dan menghapus bila terdapat nilai *null* maupun *missing value* pada baris data.

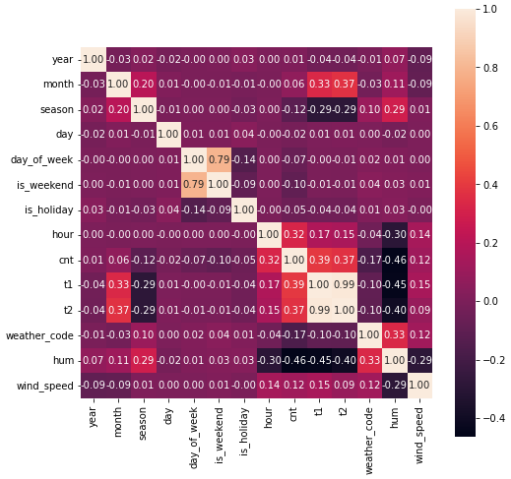
```
In [6]: # data checking (missing values/null)

        print(london.isnull().sum())
        print(london.isna().sum())

        print(london_time.isnull().sum())
        print(london_time.isna().sum())
```

```
timestamp       0
cnt             0
t1              0
t2              0
hum             0
wind_speed      0
weather_code    0
is_holiday      0
is_weekend      0
season          0
dtype: int64
timestamp       0
cnt             0
t1              0
t2              0
hum             0
wind_speed      0
weather_code    0
is_holiday      0
is_weekend      0
season          0
dtype: int64
year            0
month           0
season          0
day             0
day_of_week     0
is_weekend      0
is_holiday      0
hour            0
cnt             0
t1              0
t2              0
weather_code    0
hum             0
wind_speed      0
dtype: int64
year            0
month           0
season          0
day             0
day_of_week     0
is_weekend      0
is_holiday      0
hour            0
cnt             0
t1              0
t2              0
weather_code    0
hum             0
wind_speed      0
dtype: int64
```

Gambar 10. *Data Checking (null/NA)*

Pada data yang digunakan, tidak ditemukan nilai *null* dan NA sehingga kita dapat langsung masuk ke proses berikutnya, yakni pemodelan.

Proses berikutnya adalah mencari korelasi dengan *figure heatmap* terhadap target variabel dengan variabel pendukung lain untuk digunakan sebagai model.



Gambar 11. *Heatmap Figure*

Dari korelasi yang tervisualisasi pada figur *heatmap* tersebut, dapat terlihat bahwa pada target variabel "is_holiday" memuat 4 variabel yang memiliki korelasi tinggi di atas 0, yakni "year", "day", "weather_code", dan "hum". Untuk 9 variabel lainnya, memiliki nilai korelasi di bawah 0. Sedangkan, pada target variabel "is_weekend" memuat "day", "day_of_week", "weather_code", "hum", dan "wind_speed" dengan total 5 variabel yang memiliki korelasi tinggi di atas 0 dan sisa 8 variabel lainnya memiliki korelasi di bawah 0.

d. Modeling

- *Naive Bayes*
  - *is_holiday*

Model pertama dengan menggunakan klasifikasi Naive Bayes dengan target variabel “is_holiday”. Langkah pertama yang dilakukan adalah membagi data ke *training* dan *testing* data dengan rasio 70:30 menggunakan fungsi train_test_split.

```
# naive bayes model

# is_holiday
# split: training and testing
from sklearn.model_selection import train_test_split

#membagi data ke X dan Y

london_model = london_time.copy()
Y = london_model['is_holiday']
X = london_model.drop(columns='is_holiday')

#membagi data training 70% dan testing 30%
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
```

Gambar 12. *Data Split (is_holiday)*

Setelah data terbagi, dilakukan *model fitting* menggunakan fungsi GaussianNB pada *library*, menggunakan data yang telah terbagi menjadi 2 model *training* dan *testing* untuk dilakukannya pengujian akurasi terhadap kedua data. Pada *data training*, didapatkan akurasi sebesar 84% dan *data testing* sebesar 83%.

```
# naive bayes

from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'.format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'.format(gnb.score(X_test, y_test)))
```

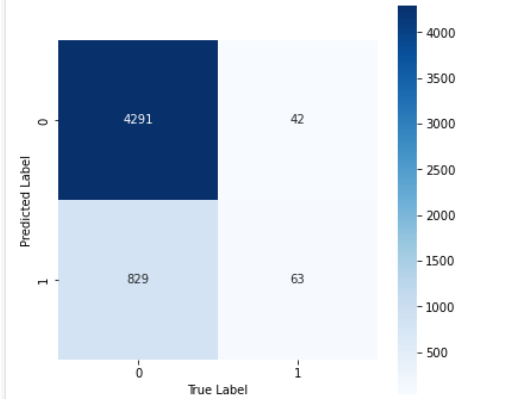
```
Accuracy of GNB classifier on training set: 0.84
Accuracy of GNB classifier on test set: 0.83
```

Gambar 13. Akurasi Model Naive Bayes *(is_holiday)*

Untuk mengetahui tingkat ketepatan dan keberhasilan dari suatu model prediktif, *Confusion Matrix* disediakan sebagai alat pengujian untuk menemukan prediksi yang benar maupun salah.

Naive Bayes Classifier

	precision	recall	f1-score	support
0	0.99	0.84	0.91	5120
1	0.07	0.60	0.13	105
accuracy			0.83	5225
macro avg	0.53	0.72	0.52	5225
weighted avg	0.97	0.83	0.89	5225



Gambar 14. Confusion Matrix Model Naive Bayes *(is_holiday)*

Dari hasil *Confusion Matrix* tersebut, didapatkan 4.291 *True Positive* dan 42 *False Positive*, serta 829 *False Negative* dan 63 *True Negative*. Kemudian, kelompok kami juga melakukan *Cross Validation* dengan menggunakan *library* cross_val_score untuk menemukan akurasi yang maksimal dari model.

```
from sklearn.model_selection import cross_val_score

scores1 = cross_val_score(gnb, X_train, y_train, cv=10)

#Cross Validation naive bayes

print('Cross-Validation Accuracy Scores for Naive Bayes', scores1)
scores1 = pd.Series(scores1)
scores1.min(), scores1.mean(), scores1.max()
```

```
Cross-Validation Accuracy Scores for Naive Bayes [0.8269073  0.83100902 0.86710418 0.8490566  0.80885972 0.85808039
 0.83839212 0.83839212 0.82608696 0.83743842]

(0.8088597210828548, 0.838132685678724, 0.867104183757178)
```

Gambar 15. *Cross Validation* Model Naive Bayes *(is_holiday)*

Berdasarkan pengujian *Cross Validation* yang dilakukan diperoleh hasil dengan akurasi terendah 80%, rata-rata akurasi 83% dan akurasi tertinggi yaitu 86%.

- *is_weekend*

Model kedua dengan menggunakan klasifikasi Naive Bayes dengan target variabel “is_weekend”. Langkah pertama yang dilakukan adalah membagi data ke *training* dan *testing* data dengan rasio 70:30 menggunakan fungsi train_test_split.

```
#is_weekend

# split: training and testing
#membagi data ke X dan Y

london_model2 = london_time.copy()
Y2 = london_model2['is_weekend']
X2 = london_model2.drop(columns='is_weekend')

#membagi data training 70% dan testing 30%
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, Y2, test_size=0.3, random_state=0)
```

Gambar 16. *Data Split (is_weekend)*

Setelah data terbagi, dilakukan *model fitting* menggunakan fungsi GaussianNB pada *library*, menggunakan data yang telah terbagi menjadi 2 model *training* dan *testing* untuk dilakukannya pengujian akurasi terhadap kedua data. Pada *data training*, didapatkan akurasi sebesar 84% dan *data testing* sebesar 83%.

```
# naive bayes

gnb2 = GaussianNB()
gnb2.fit(X2_train, y2_train)
print('Accuracy of GNB classifier on training set: {:.2f}'.format(gnb2.score(X2_train, y2_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'.format(gnb2.score(X2_test, y2_test)))
```

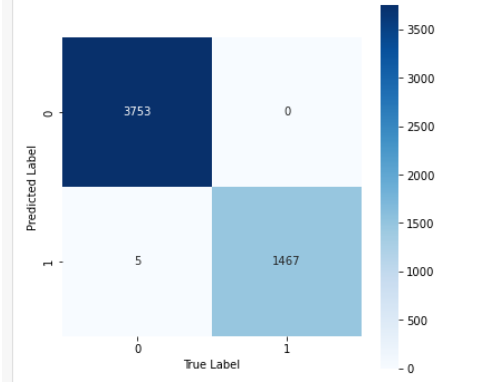
```
Accuracy of GNB classifier on training set: 1.00
Accuracy of GNB classifier on test set: 1.00
```

Gambar 17. Akurasi Model Naive Bayes *(is_holiday)*

Lalu dilakukan pencarian *Confusion Matrix* untuk menemukan prediksi yang benar dan salah.

Naive Bayes Classifier

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3758
1	1.00	1.00	1.00	1467
accuracy			1.00	5225
macro avg	1.00	1.00	1.00	5225
weighted avg	1.00	1.00	1.00	5225



Gambar 18. *Confusion Matrix* Model Naive Bayes *(is_weekend)*

Berdasarkan hasil *Confusion Matrix* di atas, didapatkan 3.753 *True Positive* dan 0 *False Positive*, serta 5 *False Negative* dan 1.467 *True Negative*.

Pengujian juga didukung dengan melakukan *Cross Validation* menggunakan *library* cross_val_score untuk menemukan akurasi yang maksimal dari model yang sedang diuji.

```
scores2 = cross_val_score(gnb2, X2_train, y2_train, cv=10)

# Cross Validation naive bayes

print('Cross-Validation Accuracy Scores for Naive Bayes', scores2)
scores2 = pd.Series(scores2)
scores2.min(), scores2.mean(), scores2.max()
```

```
Cross-Validation Accuracy Scores for Naive Bayes [1.         0.99835931 0.99917966 1.         0.99917966 1.
 0.99917966 0.99917966 0.99917966 1.        ]

(0.9983593109105825, 0.999425758818704, 1.0)
```

Gambar 19. *Cross Validation* Model Naive Bayes *(is_weekend)*

Berdasarkan pengujian *Cross Validation* yang dilakukan diperoleh hasil dengan akurasi terendah sebesar 99,8%, rata-rata akurasi sebesar 99,9% dan akurasi tertinggi yaitu 100%.

- *Random Forest*
  - *is_holiday*

Model dari penelitian kami selanjutnya yaitu menggunakan metode Random Forest, di mana target variabelnya yaitu “is_holiday”. Langkah pertama yang dilakukan sebelum mengolah data ini lebih lanjut dengan melakukan pengelompokkan kolom yang ingin digunakan untuk pemodelan nantinya. Kemudian, dilakukan *split data training* dan *test* dengan rasio 70:30. Setelah melakukan *data training* dan *testing*, kami membuat sebuah model Random Forest dengan *library* RandomForestClassifier dengan penamaan objek “rf”.

```
# random forest model

#is_holiday
#membagi X dan y
feature_cols = ['cnt','t1', 't2','hum','wind_speed','year','month','season','d
X1 = london_time[feature_cols] #features
y1 = london_time['is_holiday'] #target variable
```

```
#split dataset into training and test set

from sklearn.model_selection import train_test_split

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.3,
```

```
from sklearn.ensemble import RandomForestClassifier

#create decision tree classifier object
rf = RandomForestClassifier()

#train decision tree classifier
rf = rf.fit(X1_train,y1_train)

#predict the response for test dataset
y1_pred_rf = rf.predict(X1_test)
```

Gambar 20. Pemodelan Random Forest *(is_holiday)*

Untuk mengetahui tingkat ketepatan dalam model ini, maka kami menggunakan *Confusion Matrix* yang disediakan sebagai salah satu bentuk dari pengujian untuk menentukan prediksi mana yang benar atau salah.

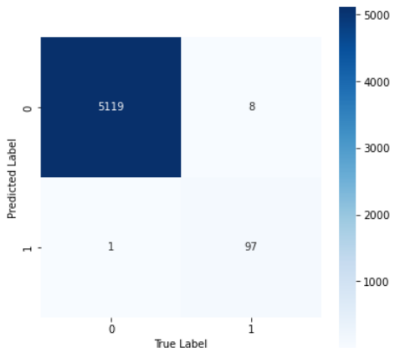
```
#Confusion Matrix

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y1_pred = rf.predict(X1_test)


mat1 = confusion_matrix(y1_test, y1_pred)
plt.figure(figsize=(6,6))
sns.heatmap(mat1.T, square= True, annot= True, cmap='Blues', fmt="d")
plt.xlabel('True Label')
plt.ylabel('Predicted Label')

# precision, recall f1-score and accuracy
print("Random Forest Classifier\n")
print(classification_report(y1_test, y1_pred))
```

Gambar 21. *Confusion Matrix* Model Random Forest *(is_holiday)*

Berikut merupakan hasil dari Confusion Matrix pada variabel is_holiday:

Random Forest Classifier

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5120
1	0.99	0.92	0.96	105
accuracy			1.00	5225
macro avg	0.99	0.96	0.98	5225
weighted avg	1.00	1.00	1.00	5225



Gambar 22. *Confusion Matrix* Model Random Forest *(is_holiday)* (2)

Berdasarkan hasil *Confusion Matrix* di atas, dapat disimpulkan bahwa kami mendapatkan

5.119 *True Positive* dan 8 *False Positive*, serta 1 *False Negative* dan 97 *True Negative*.

Setelah melakukan *Confusion Matrix*, tahap selanjutnya, kami mencari akurasi dari model yang kami gunakan pada variabel terkait, di mana dalam proses ini dilakukan *Cross Validation* dalam pencarian akurasinya. Berikut merupakan tahap pencarian akurasi dengan *Cross Validation*:

```
#Model accuracy
from sklearn import metrics

print("Accuracy Random Forest:",metrics.accuracy_score(y1_test,y1_pred_rf))
```

```
Accuracy Random Forest: 0.9982775119617225
```

```
#Cross Validation Random Forest

from sklearn.model_selection import cross_val_score

scores2 = cross_val_score(rf, X1_train, y1_train, cv=10)

#Cross Validation

print('Cross-Validation Accuracy Scores for Random Forest', scores2)
scores2 = pd.Series(scores2)
scores2.min(), scores2.mean(), scores2.max()
```

```
Cross-Validation Accuracy Scores for Random Forest [0.99835931 0.99753897
1.         0.99753897 0.99753897 0.99835931
 0.99835931 0.99835931 1.         0.99835796]

(0.9975389663658737, 0.9984412106615157, 1.0)
```

Gambar 23. Akurasi dan *Cross Validation* Model Random Forest *(is_holiday)*

Dari hasil tersebut didapatkan hasil akurasi model Random Forest untuk variabel "is_holiday" sebesar 99% dan setelah dilakukan Cross Validation, nilai maksimum berubah ke angka 100% dan nilai minimum yaitu 99%.

- *is_weekend*

Pada target variabel kedua yaitu "is_weekend", sama dengan tahap yang dilakukan dalam pengolahan data pada variabel is_holiday, di mana kami melakukan pengelompokkan kolom yang ingin digunakan untuk pemodelan nantinya. Kemudian, dilakukan *split data training* dan *testing*. Setelah melakukan *data training* dan *testing*, kami membuat sebuah *library* baru yang bernama RandomForestClassifier dengan penamaan objek "rf".

```
#random forest model

#is_weekend
#membagi X dan y
feature_cols = ['cnt','t1', 't2','hum','wind_speed','year','month','season
X3 = london_time[feature_cols] #features
y3 = london_time['is_weekend'] #target variable
```

```
#split dataset into training and test set

X1_train, X1_test, y1_train, y1_test = train_test_split(X3, y3, test_size=
```

```
#create decision tree classifier object
rf = RandomForestClassifier()

#train decision tree classifier
rf = rf.fit(X1_train,y1_train)

#predict the response for test dataset
y1_pred_rf = rf.predict(X1_test)
```

Gambar 24. Pemodelan Random Forest *(is_weekend)*

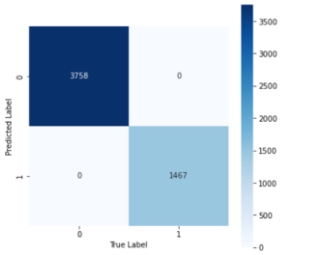
Untuk mengetahui tingkat ketepatan dalam model ini, maka kami kembali menggunakan *Confusion Matrix* yang disediakan sebagai salah satu bentuk dari pengujian untuk menentukan prediksi mana yang benar atau salah.

```
#Confusion Matrix

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y1_pred = rf.predict(X1_test)


mat1 = confusion_matrix(y1_test, y1_pred)
plt.figure(figsize=(6,6))
sns.heatmap(mat1.T, square= True, annot= True, cmap='Blues', fmt="d")
plt.xlabel('True Label')
plt.ylabel('Predicted Label')

# precision, recall f1-score and accuracy
print("Random Forest Classifier\n")
print(classification_report(y1_test, y1_pred))
```

Gambar 25. *Confusion Matrix* Model Random Forest *(is_weekend)*

Berikut merupakan hasil dari Confusion Matrix pada variabel "is_weekend".

```
Random Forest Classifier

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3758
           1       1.00      1.00      1.00      1467

    accuracy                           1.00      5225
   macro avg       1.00      1.00      1.00      5225
weighted avg       1.00      1.00      1.00      5225
```



Gambar 26. *Confusion Matrix* Model Random Forest *(is_weekend)* (2)

Berdasarkan hasil *Confusion Matrix* di atas, dapat disimpulkan bahwa kami mendapatkan 3.758 *True Positive* dan 0 *False Positive*, serta 0 *False Negative* dan 1.467 *True Negative*.

Setelah melakukan *Confusion Matrix*, kami mencari akurasi dari model yang kami gunakan pada variabel terkait, di mana dalam proses ini dilakukan *Cross Validation* dalam pencarian akurasinya. Berikut merupakan tahap pencarian akurasi dengan *Cross Validation*.

```
#Model accuracy
from sklearn import metrics

print("Accuracy Random Forest:",metrics.accuracy_score(y1_test,y1_pred_rf))
```

```
Accuracy Random Forest: 1.0
```

```
#Cross Validation Random Forest

from sklearn.model_selection import cross_val_score

scores2 = cross_val_score(rf, X1_train, y1_train, cv=10)

#Cross Validation

print('Cross-Validation Accuracy Scores for Random Forest', scores2)
scores2 = pd.Series(scores2)
scores2.min(), scores2.mean(), scores2.max()
```

```
Cross-Validation Accuracy Scores for Random Forest [1. 1. 1. 1. 1. 1. 1.
1. 1. 1.]

(1.0, 1.0, 1.0)
```

Gambar 27. Akurasi dan *Cross Validation* Model Random Forest *(is_weekend)*

Dari hasil di atas, kami mendapatkan tidak ada perubahan setelah dilakukan *Cross Validation* di mana akurasi dari pemodelan pada variabel "is_weekend" tetap pada angka 100%.

e. Evaluation

Pada tahap *Evaluation*, kelompok kami melakukan komparasi dari kedua model untuk menemukan model mana yang menghasilkan *output* terbaik. Pengujian diawali dengan melakukan komparasi terhadap akurasi dari model sebelum dan sesudah *Cross Validation*. Pada model Naive Bayes dengan target variabel "is_holiday", rata-rata akurasi sebelum proses *Cross Validation* adalah 83% dan setelah dilakukan proses *Cross Validation* berubah menjadi 86% untuk akurasi maksimumnya.

```
Accuracy of GNB classifier on training set: 0.84
Accuracy of GNB classifier on test set: 0.83

Cross-Validation Accuracy Scores for Naive Bayes [0.8269073
0.83100902 0.86710418 0.8490566  0.80885972 0.85808039
 0.83839212 0.83839212 0.82608696 0.83743842]

(0.8088597210828548, 0.838132685678724, 0.867104183757178)
```

Gambar 28. Akurasi dan *Cross Validation* Model Naive Bayes dari *is_holiday*

Sedangkan pada model Naive Bayes dengan target variabel "is_weekend", rata-rata akurasi sebelum proses *Cross Validation* adalah 100% dan setelah dilakukan proses *Cross Validation* berubah menjadi 100% untuk akurasi maksimumnya.

```
Accuracy of GNB classifier on training set: 1.00
Accuracy of GNB classifier on test set: 1.00

Cross-Validation Accuracy Scores for Naive Bayes [1.
0.99835931 0.99917966 1.         0.99917966 1.
 0.99917966 0.99917966 0.99917966 1.        ]

(0.9983593109105825, 0.999425758818704, 1.0)
```

Gambar 29. Akurasi dan *Cross Validation* Model Naive Bayes dari *is_weekend*

Untuk model klasifikasi selanjutnya, model menggunakan algoritma Random Forest dengan melakukan komparasi yang sama. Sebelum dilakukan *Cross Validation* pada target variabel "is_holiday", model kami mendapatkan akurasi sebesar 99% dan setelah dilakukan *Cross Validation* akurasinya berubah menjadi 100% untuk akurasi maksimum.

```
Accuracy Random Forest: 0.9982775119617225

Cross-Validation Accuracy Scores for Random Forest [0.998359
31 0.99753897 1.         0.99753897 0.99753897 0.99835931
 0.99835931 0.99835931 1.         0.99835796]

(0.9975389663658737, 0.9984412106615157, 1.0)
```

Gambar 30. Akurasi dan *Cross Validation* Model Random Forest dari *is_holiday*

Sedangkan pada model Random Forest dengan target variabel "is_weekend", akurasi sebelum dan sesudah proses *Cross Validation* adalah 100%.

```
Accuracy Random Forest: 1.0

Cross-Validation Accuracy Scores for Random Forest [1. 1. 1.
1. 1. 1. 1. 1. 1. 1.]

(1.0, 1.0, 1.0)
```

Gambar 30. Akurasi dan *Cross Validation* Model Random Forest dari *is_weekend*

Jika membandingkan *Confusion Matrix* kedua model pada target variabel "is_holiday", kita bisa melihat bahwa Naive Bayes dapat memprediksi 4.291 *True Positive*, sedangkan Random Forest sebesar 5.119. Namun untuk Random Forest sendiri berhasil memprediksi *False Positive* yang lebih sedikit sehingga lebih akurat. Sedangkan pada perbandingan *Confusion Matrix* kedua model pada target variabel "is_weekend", Naive Bayes dapat memprediksi 3.753 *True Positive*, sedangkan Random Forest sebesar 3.758.

Jika melihat semua perbandingan yang telah dilakukan, kami memilih Random Forest sebagai model terbaik untuk analisis kami karena memiliki akurasi yang lebih tinggi dari Naive Bayes, baik sebelum Cross Validation maupun setelah Cross Validation yang mencapai akurasi sempurna di 100%

f. Deployment

Dari hasil yang diperoleh dari pemodelan, dapat disimpulkan bahwa terdapat penggunaan *bike-sharing* sebesar 5.119 orang yang akan bersepeda di hari bukan libur dan 3.758 orang akan bersepeda bukan pada hari akhir minggu. Untuk saran dari hasil analisis kami, diperlukan konsistensi dan improvisasi terhadap penunjang infrastuktur dan sistem yang sudah dibuat agar pengguna transportasi umum dan penyewaan merasa lebih nyaman pada pelayanan bersama di bidang transportasi umum.

V. CONCLUSION

Dari hasil percobaan yang dilakukan terhadap *dataset* yang sudah ditentukan, dengan menggunakan dua metode untuk mengolah data tersebut yaitu Naive Bayes dan Random Forest pada variabel is_weekend dan is_holiday untuk memprediksi tingkat penyewaan yang terjadi di kota London pada hari libur ataupun hari pekan. Kedua metode tersebut menghasilkan hasil yang berbeda, di mana pada kasus hari libur yakni diwakilkan sebagai is_holiday pada *dataset*, Dilansir berdasarkan hasil dari kedua algoritma tersebut, menghasilkan kisaran sebesar 5.119 orang yang akan bersepeda pada hari yang bukan pada hari libur, sedangkan berdasarkan klasifikasi Random Forest dengan hasil akurasi yaitu sebesar 100% memiliki

akurasi yang lebih besar apabila dibandingkan dengan metode Naive Bayes yang hanya menghasilkan tingkat akurasi yaitu sebesar 86%. Pada kasus akhir minggu, yakni is_weekend sebagai variabel pada *dataset*, dilansir dari hasil kedua algoritma tersebut, menghasilkan kisaran sebesar 3.758 orang yang akan bersepeda bukan di akhir minggu. Sedangkan, berdasarkan klasifikasi Random Forest, dengan tingkat akurasi yaitu sebesar 100%, menghasilkan hasil akurasi yang sedikit lebih tinggi dibandingkan metode klasifikasi Naive Bayes dengan tingkat akurasi yaitu sebesar 99%. Melalui data yang kita dapat dan hasil dari olahan data menggunakan kedua algoritma tersebut, dapat ditarik kesimpulan bahwa model Random Forest lebih cocok untuk digunakan dalam mengklasifikasi dan prediksi *dataset* penelitian ini, dengan hasil akurasi pada kedua variabel yaitu sebesar 100%.

Melalui penelitian ini, diharapkan akan ada penelitian berkelanjutan dimana penelitian ini bisa dijadikan sebagai referensi untuk penelitian kedepannya dalam penggunaan algoritma Naive Bayes maupun Random Forest, serta adanya pola-pola atau teknik baru untuk mengolah data *London Bike Sharing*, untuk menghasilkan data lebih yang bisa berguna untuk peneliti lainnya, serta dapat menjadi perbandingan yang bermanfaat dalam meneliti masalah atau pemecahan masalah yang baru dalam menggunakan algoritma Naives Bayes maupun Random Forest.

```
timestamp       datetime64[ns]
cnt                      int64
t1                     float64
t2                     float64
hum                    float64
wind_speed             float64
weather_code             int64
is_holiday               int64
is_weekend               int64
season                   int64
dtype: object
<bound method NDFrame.describe of                timestamp   cnt   t1   t2    hum  wind_speed  weather_code  \
0     2015-01-04 00:00:00   182  3.0  2.0   93.0         6.0             3
1     2015-01-04 01:00:00   138  3.0  2.5   93.0         5.0             1
2     2015-01-04 02:00:00   134  2.5  2.5   96.5         0.0             1
3     2015-01-04 03:00:00    72  2.0  2.0  100.0         0.0             1
4     2015-01-04 04:00:00    47  2.0  0.0   93.0         6.5             1
...                   ...   ...  ...  ...    ...         ...           ...
17409 2017-01-03 19:00:00  1042  5.0  1.0   81.0        19.0             3
17410 2017-01-03 20:00:00   541  5.0  1.0   81.0        21.0             4
17411 2017-01-03 21:00:00   337  5.5  1.5   78.5        24.0             4
17412 2017-01-03 22:00:00   224  5.5  1.5   76.0        23.0             4
17413 2017-01-03 23:00:00   139  5.0  1.0   76.0        22.0             2

       is_holiday  is_weekend  season
0               0           1       3
1               0           1       3
2               0           1       3
3               0           1       3
4               0           1       3
...           ...         ...     ...
17409           0           0       3
17410           0           0       3
17411           0           0       3
17412           0           0       3
17413           0           0       3
```

```
# duplicate dataframe
london_time = london.copy()

# specific timestamp dtype into columns
london_time['year'] = london_time['timestamp'].dt.year
london_time['month'] = london_time['timestamp'].dt.month
london_time['day'] = london_time['timestamp'].dt.day
london_time['day_of_week'] = london_time['timestamp'].dt.dayofweek
london_time['day_of_week'] += 1 #changing the index to map monday as 1 instead of 0
london_time['hour'] = london_time['timestamp'].dt.hour

london_df = ['year', 'month', 'season', 'day', 'day_of_week', 'is_weekend', 'is_holiday',
             'hour', 'cnt', 't1', 't2', 'weather_code', 'hum', 'wind_speed']
london_time = london_time[london_df]

london_time
```

	year	month	season	day	day_of_week	is_weekend	is_holiday	hour	cnt	t1	t2	weather_code	hum	wind_speed
0	2015	1	3	4	7	1	0	0	182	3.0	2.0	3	93.0	6.0
1	2015	1	3	4	7	1	0	1	138	3.0	2.5	1	93.0	5.0
2	2015	1	3	4	7	1	0	2	134	2.5	2.5	1	96.5	0.0
3	2015	1	3	4	7	1	0	3	72	2.0	2.0	1	100.0	0.0
4	2015	1	3	4	7	1	0	4	47	2.0	0.0	1	93.0	6.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17409	2017	1	3	3	2	0	0	19	1042	5.0	1.0	3	81.0	19.0
17410	2017	1	3	3	2	0	0	20	541	5.0	1.0	4	81.0	21.0
17411	2017	1	3	3	2	0	0	21	337	5.5	1.5	4	78.5	24.0
17412	2017	1	3	3	2	0	0	22	224	5.5	1.5	4	76.0	23.0
17413	2017	1	3	3	2	0	0	23	139	5.0	1.0	2	76.0	22.0

17414 rows × 14 columns

LAMPIRAN

Group 5

- Aljevan Komala (00000044016)
- Jerrell Susilo (00000045370)
- Thomas Januardy (00000046001)

```
# import library

import pandas as pd
import numpy as py
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# read data

london = pd.read_csv("london_merged.csv", sep=",")
london.info()
london
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17414 entries, 0 to 17413
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   timestamp     17414 non-null  object
 1   cnt           17414 non-null  int64
 2   t1            17414 non-null  float64
 3   t2            17414 non-null  float64
 4   hum           17414 non-null  float64
 5   wind_speed    17414 non-null  float64
 6   weather_code  17414 non-null  float64
 7   is_holiday    17414 non-null  float64
 8   is_weekend    17414 non-null  float64
 9   season        17414 non-null  float64
dtypes: float64(8), int64(1), object(1)
memory usage: 1.3+ MB
```

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0
...	...	...	...	...	...	...	...	...	...	...
17409	2017-01-03 19:00:00	1042	5.0	1.0	81.0	19.0	3.0	0.0	0.0	3.0
17410	2017-01-03 20:00:00	541	5.0	1.0	81.0	21.0	4.0	0.0	0.0	3.0
17411	2017-01-03 21:00:00	337	5.5	1.5	78.5	24.0	4.0	0.0	0.0	3.0
17412	2017-01-03 22:00:00	224	5.5	1.5	76.0	23.0	4.0	0.0	0.0	3.0
17413	2017-01-03 23:00:00	139	5.0	1.0	76.0	22.0	2.0	0.0	0.0	3.0

17414 rows × 10 columns

```
# data preprocessing

# convert dtypes
london = london.astype({'weather_code':'int64'})
london = london.astype({'is_holiday':'int64'})
london = london.astype({'is_weekend':'int64'})
london = london.astype({'season':'int64'})

london['timestamp'] = pd.to_datetime(london['timestamp'])

print(london.dtypes)
print(london.describe)
```

```
# data checking (missing values/null)

print(london.isnull().sum())
print(london.isna().sum())

print(london_time.isnull().sum())
print(london_time.isna().sum())
```

```
timestamp       0
cnt             0
t1              0
t2              0
hum             0
wind_speed      0
weather_code    0
is_holiday      0
is_weekend      0
season          0
dtype: int64
timestamp       0
cnt             0
t1              0
t2              0
hum             0
wind_speed      0
weather_code    0
is_holiday      0
is_weekend      0
season          0
dtype: int64
year            0
month           0
season          0
day             0
day_of_week     0
is_weekend      0
is_holiday      0
hour            0
cnt             0
```

```
# data group by target variables (is_holiday)
london.groupby('is_holiday').describe().T
```

is_holiday		0	1
cnt	count	17030.000000	384.000000
	mean	1151.525191	799.526042
	std	1089.182222	802.242181
	min	0.000000	14.000000
	25%	261.000000	171.500000
...	...	...	...
season	min	0.000000	0.000000
	25%	1.000000	0.000000
	50%	1.000000	0.500000
	75%	2.000000	3.000000
	max	3.000000	3.000000

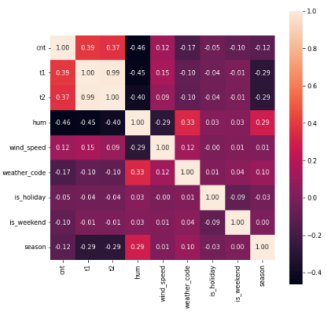
64 rows × 2 columns

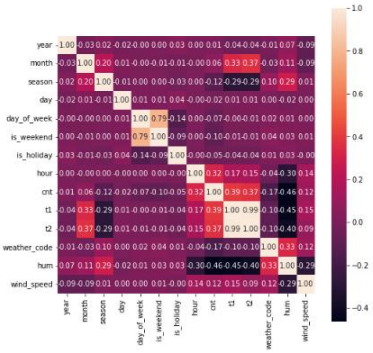
```
# data group by target variables (is_weekend)
london.groupby('is_weekend').describe().T
```

is_weekend		0	1
cnt	count	12444.000000	4970.000000
	mean	1209.274831	977.415694
	std	1136.037077	925.205089
	min	9.000000	0.000000
	25%	258.000000	256.000000
...	...	...	...
	min	0.000000	0.000000
	25%	0.000000	0.000000
	50%	1.000000	1.000000

```
# heatmap figure
plt.figure(figsize=(8,8))
sns.heatmap(london.corr(), square=True, annot=True, fmt='1.2f')

# heatmap figure (time)
plt.figure(figsize=(8,8))
sns.heatmap(london_time.corr(), square=True, annot=True, fmt='1.2f')
```

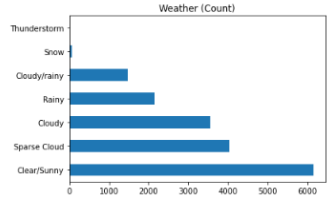
```
<AxesSubplot:>
```

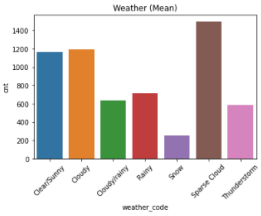


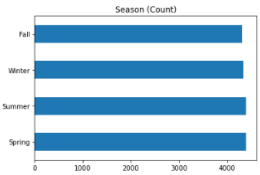


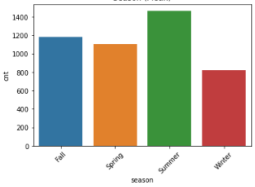

```
# data visualization

london_viz = london_time.copy()

# weather
weather_map = {1: 'Clear/Sunny', 2:'Sparse Cloud', 3:'Cloudy', 4:'Cloudy/rainy', 7:'Rainy', 10:'Thunderstorm', 26:"Snow"}
london_viz['weather_code'] = london_viz['weather_code'].map(weather_map)

london_weather = london_viz['weather_code'].value_counts()
london_weather.plot(kind='barh')
plt.title('Weather (Count)')
plt.show()

london_weather_mean = london_viz.groupby(['weather_code'])['cnt'].mean().reset_index()
sns.barplot(data=london_weather_mean, x='weather_code', y='cnt')
plt.xticks(rotation=45)
plt.title('Weather (Mean)')
plt.show()

# season
season_map = {0: 'Spring', 1:'Summer', 2:'Fall', 3:'Winter'}
london_viz['season'] = london_viz['season'].map(season_map)

london_season = london_viz['season'].value_counts()
london_season.plot(kind='barh')
plt.title('Season (Count)')
plt.show()

london_season_mean = london_viz.groupby(['season'])['cnt'].mean().reset_index()
sns.barplot(data=london_season_mean, x='season', y='cnt')
plt.xticks(rotation=45)
plt.title('Season (Mean)')
plt.show()
```

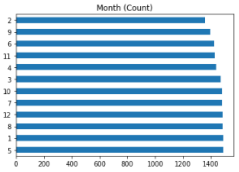
```
# month
london_month = london_viz['month'].value_counts()
london_month.plot(kind='barh')
plt.title('Month (Count)')
plt.show()

london_month_mean = london_viz.groupby(['month'])['cnt'].mean().reset_index()
sns.barplot(data=london_month_mean, x='month', y='cnt')
plt.xticks(rotation=45)
plt.title('Month (Mean)')
plt.show()
```

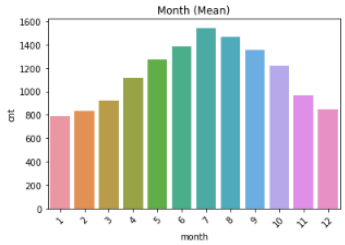




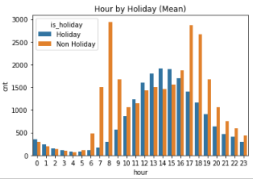




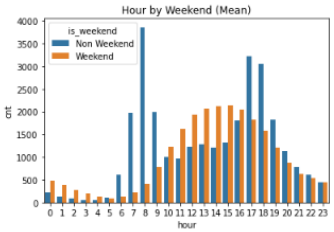




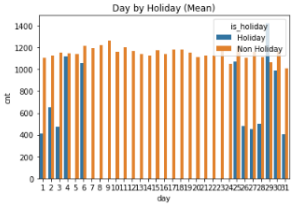

```
# data visualization

# hour by holiday (mean)
holiday_map = {0: 'Non Holiday', 1:'Holiday'}
london_viz['is_holiday'] = london_viz['is_holiday'].map(holiday_map)

london_holiday = london_viz.groupby(['hour', 'is_holiday'])['cnt'].mean().reset_index()
sns.barplot(data=london_holiday, x='hour', y='cnt', hue='is_holiday')
plt.title("Hour by Holiday (Mean)")
plt.show()
```



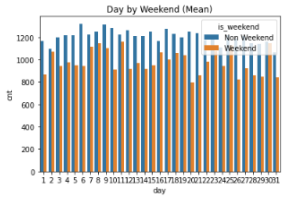

```
# hour by weekend (mean)
weekend_map = {0: 'Non Weekend', 1:'Weekend'}
london_viz['is_weekend'] = london_viz['is_weekend'].map(weekend_map)

london_weekend = london_viz.groupby(['hour', 'is_weekend'])['cnt'].mean().reset_index(
sns.barplot(data=london_weekend, x='hour', y='cnt', hue='is_weekend')
plt.title("Hour by Weekend (Mean)")
plt.show()
```



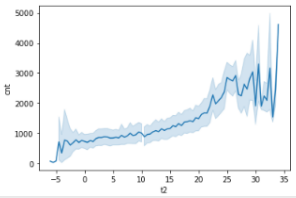

```
# day by holiday (mean)

london_holiday2 = london_viz.groupby(['day', 'is_holiday'])['cnt'].mean().reset_index()
sns.barplot(data=london_holiday2, x='day', y='cnt', hue='is_holiday')
plt.title("Day by Holiday (Mean)")
plt.show()
```




```
# day by weekend (mean)

london_weekend2 = london_viz.groupby(['day', 'is_weekend'])['cnt'].mean().reset_index(
sns.barplot(data=london_weekend2, x='day', y='cnt', hue='is_weekend')
plt.title("Day by Weekend (Mean)")
plt.show()
```




```
# temperature
london_temp = london_viz.groupby(['hour', 't2'])['cnt'].mean().reset_index()
sns.lineplot(data=london_temp, x='t2', y='cnt')
plt.show()
```




```
# naive bayes model

# is_holiday
# split: training and testing
from sklearn.model_selection import train_test_split

#membagi data ke X dan Y

london_model = london_time.copy()
Y = london_model['is_holiday']
X = london_model.drop(columns='is_holiday')

#membagi data training 70% dan testing 30%
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
```

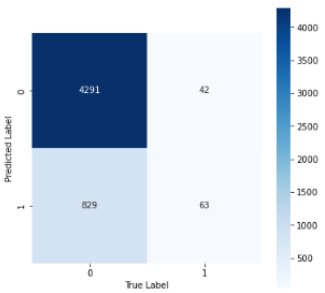
```
# naive bayes

from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'.format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'.format(gnb.score(X_test, y_test)))
```

```
Accuracy of GNB classifier on training set: 0.84
Accuracy of GNB classifier on test set: 0.83
```

```
# confusion matrix

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y_pred = gnb.predict(X_test)


mat = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,6))
sns.heatmap(mat.T, square= True, annot= True, cmap='Blues', fmt="d")
plt.xlabel('True Label')
plt.ylabel('Predicted Label')
```

```
# precision, recall f1-score and accuracy
print("Naive Bayes Classifier\n")
print(classification_report(y_test, y_pred))
```

```
Naive Bayes Classifier

              precision    recall  f1-score   support

           0       0.99      0.84      0.91      5120
           1       0.07      0.60      0.13       105

    accuracy                           0.83      5225
   macro avg       0.53      0.72      0.52      5225
weighted avg       0.97      0.83      0.89      5225
```




```
from sklearn.model_selection import cross_val_score

scores1 = cross_val_score(gnb, X_train, y_train, cv=10)
```

```
#Cross Validation naive bayes

print('Cross-Validation Accuracy Scores for Naive Bayes', scores1)
scores1 = pd.Series(scores1)
scores1.min(), scores1.mean(), scores1.max()
```

```
Cross-Validation Accuracy Scores for Naive Bayes [0.8269073  0.83100902 0.86710418 0.8490566  0.80885972 0.85080039
 0.83839212 0.83839212 0.82608696 0.83743842]

(0.8088597210828548, 0.838132685678724, 0.8671041837571778)
```

```
#is_weekend

# split: training and testing
#membagi data ke X dan Y

london_model2 = london_time.copy()
Y2 = london_model2['is_weekend']
X2 = london_model2.drop(columns='is_weekend')

#membagi data training 70% dan testing 30%
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, Y2, test_size=0.3, random_state=0)
```

```
# naive bayes

gnb2 = GaussianNB()
gnb2.fit(X2_train, y2_train)
print('Accuracy of GNB classifier on training set: {:.2f}'.format(gnb2.score(X2_train, y2_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'.format(gnb2.score(X2_test, y2_test)))
```

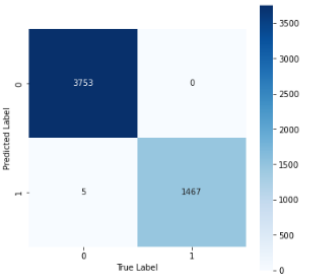
```
Accuracy of GNB classifier on training set: 1.00
Accuracy of GNB classifier on test set: 1.00
```

```
# confusion matrix
y2_pred = gnb2.predict(X2_test)


mat2 = confusion_matrix(y2_test, y2_pred)
plt.figure(figsize=(6,6))
sns.heatmap(mat2.T, square= True, annot= True, cmap='Blues', fmt="d")
plt.xlabel('True Label')
plt.ylabel('Predicted Label')

# precision, recall f1-score and accuracy
print("Naive Bayes Classifier\n")
print(classification_report(y2_test, y2_pred))
```

```
Naive Bayes Classifier
```

Naive Bayes Classifier

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3758
1	1.00	1.00	1.00	1467
accuracy			1.00	5225
macro avg	1.00	1.00	1.00	5225
weighted avg	1.00	1.00	1.00	5225




```
scores2 = cross_val_score(gnb2, X2_train, y2_train, cv=10)
```

```
# Cross Validation naive bayes

print('Cross-Validation Accuracy Scores for Naive Bayes', scores2)
scores2 = pd.Series(scores2)
scores2.min(), scores2.mean(), scores2.max()
```

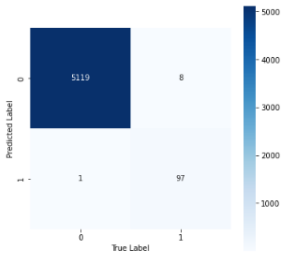
```
Cross-Validation Accuracy Scores for Naive Bayes [1.         0.99835931 0.99917966 1.         0.99917966 1.
 0.99917966 0.99917966 0.99917966 1.        ]

(0.9983593109105825, 0.9994257588818704, 1.0)
```

```
# random forest model

#is_holiday
#menbagi X dan y
feature_cols = ['cnt','t1', 't2','hum','wind_speed','year','month','season','day','day_of_week','hour','weather_code']
X1 = london_time[feature_cols] #features
y1 = london_time['is_holiday'] #target variable

#split dataset into training and test set

from sklearn.model_selection import train_test_split

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.3, random_state=0) # 70 % training and 30 % test

from sklearn.ensemble import RandomForestClassifier

#create decision tree classifier object
rf = RandomForestClassifier()

#train decision tree classifier
rf = rf.fit(X1_train,y1_train)

#predict the response for test dataset
y1_pred_rf = rf.predict(X1_test)
```

```
#Confusion Matrix

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y1_pred = rf.predict(X1_test)


mat1 = confusion_matrix(y1_test, y1_pred)
plt.figure(figsize=(6,6))
sns.heatmap(mat1.T, square= True, annot= True, cmap='Blues', fmt="d")
plt.xlabel('True Label')
plt.ylabel('Predicted Label')

# precision, recall f1-score and accuracy
print("Random Forest Classifier\n")
print(classification_report(y1_test, y1_pred))
```

Random Forest Classifier

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5120
1	0.99	0.92	0.96	105
accuracy			1.00	5225
macro avg	0.99	0.96	0.98	5225
weighted avg	1.00	1.00	1.00	5225



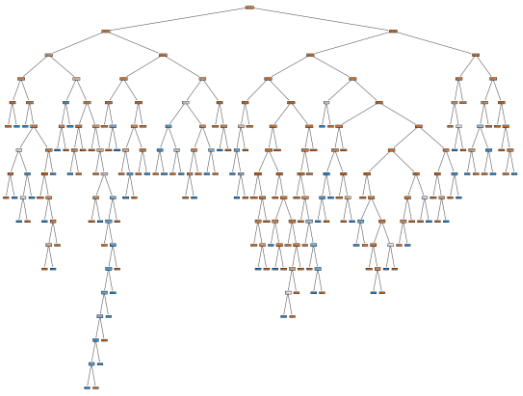

```
#Model accuracy
from sklearn import metrics

print("Accuracy Random Forest:",metrics.accuracy_score(y1_test,y1_pred_rf))
```

Accuracy Random Forest: 0.9982775119617225

```
#Cross Validation Random Forest

from sklearn.model_selection import cross_val_score

scores2 = cross_val_score(rf, X1_train, y1_train, cv=10)

#Cross Validation

print('Cross-Validation Accuracy Scores For Random Forest', scores2)
scores2 = pd.Series(scores2)
scores2.min(), scores2.mean(), scores2.max()
```

```
Cross-Validation Accuracy Scores for Random Forest [0.99835931 0.99753897 1.         0.99753897 0.99753897 0.99835931
 0.99835931 0.99835931 1.         0.99835796]

(0.9975389663658737, 0.9984412106615157, 1.0)
```

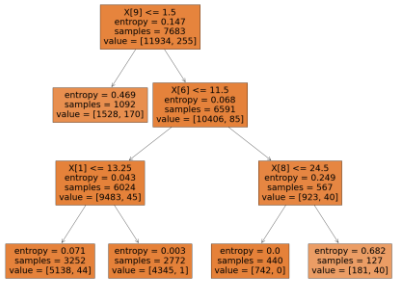
```
#without Pruning

from sklearn import tree

fig = plt.figure(figsize=(25,20))
tree.plot_tree(rf.estimators_[0],filled=True);
```



```
#with Pruning

from sklearn.ensemble import RandomForestClassifier

#create decision tree classifier obj
rf_p = RandomForestClassifier(criterion="entropy", max_depth=3, max_leaf_nodes=5, class_weight=None)

#train decision tree classifier
rf_p = rf_p.fit(X1_train,y1_train)

#predict the response for test dataset
y1_pred_rfp = rf_p.predict(X1_test)

print("Accuracy Random Forest (Pruning)",metrics.accuracy_score(y1_test,y1_pred_rfp))
```

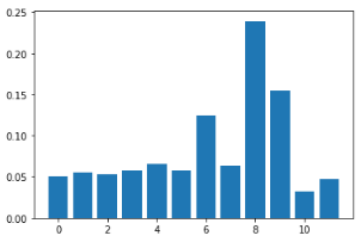
Accuracy Random Forest (Pruning) 0.9799043062200957

```
fig = plt.figure(figsize=(25,20))
tree.plot_tree(rf_p.estimators_[0],filled=True);
```




```
#Feature Importance

importance = rf.feature_importances_

for i, v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
    ""
#menampilkan plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

```
Feature: 0, Score: 0.05112
Feature: 1, Score: 0.05477
Feature: 2, Score: 0.05266
Feature: 3, Score: 0.05740
Feature: 4, Score: 0.06561
Feature: 5, Score: 0.05763
Feature: 6, Score: 0.12438
Feature: 7, Score: 0.06318
Feature: 8, Score: 0.23906
Feature: 9, Score: 0.15465
Feature: 10, Score: 0.03211
Feature: 11, Score: 0.04743
```




```
#split dataset in features and target variable
importance_cols = ['cnt', 't1', 't2','hum','wind_speed']
X2 = london_time[importance_cols] #features
y2 = london_time['is_holiday'] #target variable

#split dataset into training set and test set
X2_train, X2_test, y2_train, y2_test = train_test_split(X2,y2, test_size=0.3, random_state=0) # 70% training, 30% test
```

```
#create decision tree classifier obj
rf1 = RandomForestClassifier()

#train decision tree classifier
rf1 = rf1.fit(X2_train, y2_train)

#predict the response for test dataset
y2_pred = rf1.predict(X2_test)

print("Accuracy Random Forest (Feature Importance):",metrics.accuracy_score(y2_test,y2_pred))
```

Accuracy Random Forest (Feature Importance): 0.9799043062200957

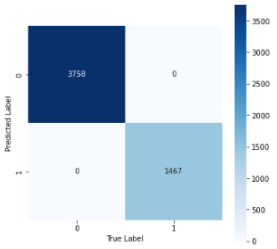
```
#random forest model

#is_weekend
#membagi X dan y
feature_cols = ['cnt','t1', 't2','hum','wind_speed','year','month','season','day','day_of_week','hour','weather_code']
X3 = london_time[feature_cols] #features
y3 = london_time['is_weekend'] #target variable
```

```
#split dataset into training and test set

X1_train, X1_test, y1_train, y1_test = train_test_split(X3, y3, test_size=0.3, random_state=0) # 70 % training and 30 % test
```

```
#create decision tree classifier object
rf = RandomForestClassifier()

#train decision tree classifier
rf = rf.fit(X1_train,y1_train)

#predict the response for test dataset
y1_pred_rf = rf.predict(X1_test)
```

```
#Confusion Matrix

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

y1_pred = rf.predict(X1_test)


mat1 = confusion_matrix(y1_test, y1_pred)
plt.figure(figsize=(6,6))
sns.heatmap(mat1.T, square= True, annot= True, cmap='Blues', fmt="d")
plt.xlabel('True Label')
plt.ylabel('Predicted Label')

# precision, recall f1-score and accuracy
print("Random Forest Classifier\n")
print(classification_report(y1_test, y1_pred))
```

```
Random Forest Classifier

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3758
           1       1.00      1.00      1.00      1467

    accuracy                           1.00      5225
   macro avg       1.00      1.00      1.00      5225
weighted avg       1.00      1.00      1.00      5225
```



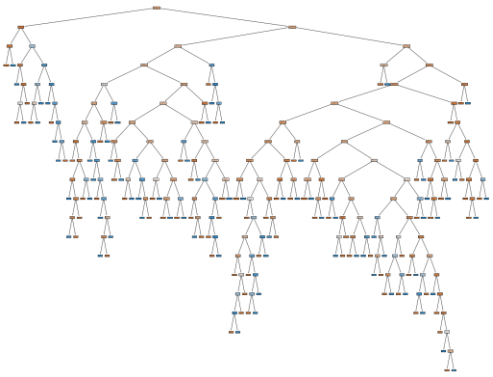

```
#Model accuracy
from sklearn import metrics

print("Accuracy Random Forest:",metrics.accuracy_score(y1_test,y1_pred_rf))
```

```
Accuracy Random Forest: 1.0
```

```
#Cross Validation Random Forest

from sklearn.model_selection import cross_val_score

scores2 = cross_val_score(rf, X1_train, y1_train, cv=10)

#Cross Validation

print('Cross-Validation Accuracy Scores for Random Forest', scores2)
scores2 = pd.Series(scores2)
scores2.min(), scores2.mean(), scores2.max()
```

```
Cross-Validation Accuracy Scores for Random Forest [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

(1.0, 1.0, 1.0)
```

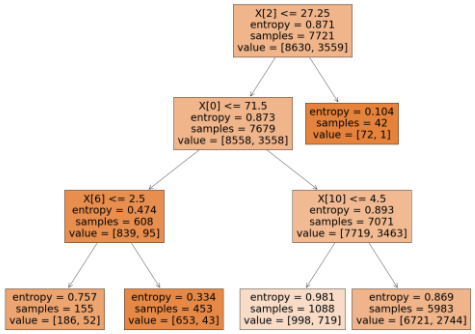
```
#Without Pruning

from sklearn import tree

fig = plt.figure(figsize=(25,20))
tree.plot_tree(rf.estimators_[0],filled=True);
```



```
#With Pruning

from sklearn.ensemble import RandomForestClassifier

#create decision tree classifier obj
rf_p = RandomForestClassifier(criterion="entropy", max_depth=3, max_leaf_nodes=5, class_weight=None)

#train decision tree classifier
rf_p = rf_p.fit(X1_train,y1_train)

#predict the response for test dataset
y1_pred_rfp = rf_p.predict(X1_test)

print("Accuracy Random Forest (Pruning)",metrics.accuracy_score(y1_test,y1_pred_rfp))
```

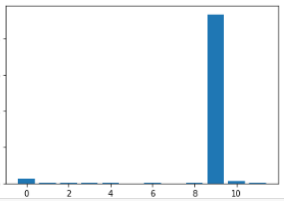
```
Accuracy Random Forest (Pruning) 1.0
```

```
fig = plt.figure(figsize=(25,20))
tree.plot_tree(rf_p.estimators_[0],filled=True);
```




```
#Feature Importance

importance = rf.feature_importances_

for i, v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
    ""
#menampilkan plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

```
Feature: 0, Score: 0.02640
Feature: 1, Score: 0.00375
Feature: 2, Score: 0.00391
Feature: 3, Score: 0.00339
Feature: 4, Score: 0.00363
Feature: 5, Score: 0.00077
Feature: 6, Score: 0.00316
Feature: 7, Score: 0.00128
Feature: 8, Score: 0.00479
Feature: 9, Score: 0.93307
Feature: 10, Score: 0.01306
Feature: 11, Score: 0.00278
```




```
#split dataset in features and target variable
importance_cols = ['cnt', 't1', 't2','hum','wind_speed']
X4 = london_time[importance_cols] #features
y4 = london_time['is_weekend'] #target variable

#split dataset into training set and test set
X2_train, X2_test, y2_train, y2_test = train_test_split(X4,y4, test_size=0.3, random_state=0) # 70% training, 30% test
```

```
#create decision tree classifier obj
rf1 = RandomForestClassifier()

#train decision tree classifier
rf1 = rf1.fit(X2_train, y2_train)

#predict the response for test dataset
y2_pred = rf1.predict(X2_test)

print("Accuracy Random Forest (Feature Importance):",metrics.accuracy_score(y2_test,y2_pred))
```

```
Accuracy Random Forest (Feature Importance): 0.7450717703349282
```

Anggota Kelompok 5

- Aljevan Komala (00000044016)

Merumuskan permasalahan bisnis dan mengembangkannya latar belakang serta literatur projek sesuai dengan metode CRISP-DM.

- Jerrell Susilo (00000045370)

Menganalisis hasil output dan menyimpulkan bisnis dengan metode CRISP-DM.

- Thomas Januardy (00000046001)

Menjalankan dan mempersiapkan data untuk pemodelan menggunakan bahasa Python pada Jupyter sesuai dengan metode CRISP-DM.

DAFTAR PUSTAKA

[1] D. Sartika and D. I. Sensuse, "Perbandingan Algoritma Klasifikasi Naive Bayes, Nearest Neighbour, dan Decision Tree pada Studi Kasus Pengambilan Keputusan Pemilihan Pola Pakaian," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 3, no. 2, pp. 151–161, Mar. 2017, doi: 10.35957/JATISI.V3I2.78.

[2] "Data mining : algoritma dan implementasi dengan pemrograman PHP / Joko Suntoro | OPAC Perpustakaan Nasional RI." https://opac.perpusnas.go.id/DetailOpac.aspx?id=1224848 (accessed Dec. 15, 2022).

[3] A. A. N. P. Redi and A. A. N. A. Redioka, "Algoritma Simulated Annealing untuk Optimasi Rute Kendaraan dan Pemindahan Lokasi Sepeda pada Sistem Public Bike Sharing," *Jurnal Sistem dan Manajemen Industri*, vol. 3, no. 1, p. 50, Jul. 2019, doi: 10.30656/JSMI.V3I1.1473.

[4] S. Beniwal and J. Arora, "Classification and Feature Selection Techniques in Data Mining," *International Journal of Engineering Research & Technology*, vol. 1, no. 6, Aug. 2012, doi: 10.17577/IJERTV1IS6124.

[5] "Implementation of Data Mining To Predict Period of Students Study Using Naive Bayes Algorithm | International Journal of Engineering and Emerging Technology." https://ojs.unud.ac.id/index.php/ijeet/article/view/34457 (accessed Dec. 15, 2022).

[6] V. W. Siburian, "Prediksi Harga Ponsel Menggunakan Metode Random Forest."

[7] B. Bustami, "PENERAPAN ALGORITMA NAIVE BAYES UNTUK MENGKLASIFIKASI DATA NASABAH ASURANSI," *TECHSI - Jurnal Teknik Informatika*, vol. 5, no. 2, Oct. 2013, doi: 10.29103/TECHSI.V5I2.154.

[8] V. Wanika Siburian, J. Sistem Komputer Universitas Sriwijaya Palembang, and I. Elvina Mulyana, "Prediksi Harga Ponsel Menggunakan Metode Random Forest," *Annual Research Seminar (ARS)*, vol. 4, no. 1, pp. 144–147, May 2019, Accessed: Dec. 15, 2022. [Online]. Available: https://seminar.ilkom.unsri.ac.id/index.php/ars/article/view/1992

[9] S. Huber, H. Wiemer, D. Schneider, and S. Ihlenfeldt, "DMME: Data mining methodology for engineering applications – a holistic extension to the CRISP-DM model," *Procedia CIRP*, vol. 79, pp. 403–408, Jan. 2019, doi: 10.1016/J.PROCIR.2019.02.106.