BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam program magang, posisi sebagai *data analyst intern* ditempatkan di bawah departemen operasional di mana Bapak Saiful Sofyan, sebagai Kepala Operasi dan supervisor, memimpin serta memberikan tugas dan proyek kepada peserta magang. Beliau juga berperan sebagai mentor yang membantu peserta magang menyelesaikan pekerjaan dan proyek selama periode magang. Koordinasi pekerjaan dilakukan baik secara langsung maupun melalui komunikasi *online* dengan rekan dari departemen operasional dan lainnya, menggunakan aplikasi WhatsApp untuk koordinasi *online*.

3.2 Tugas dan Uraian Kerja Magang

Selama menjalani program magang, peserta magang diberikan sejumlah proyek yang telah direncanakan oleh pengguna (*user*) perusahaan, masing-masing dengan tujuan yang berbeda. Selain proyek-proyek tersebut, peserta magang juga diberikan tugas-tugas harian dan berkala oleh departemen lain. Pengalaman yang didapat selama magang membantu peserta dalam mengembangkan *soft skills* serta memahami secara mendalam alur kerja perusahaan.

Tabel 3. 1 *Timeline* Rincian Pekerjaan yang Dilakukan Mahasiswa

No.	Pekerjaan yang Dilakukan	Minggu	Tanggal Mulai	Tanggal Selesai
1.	*Onboarding* perusahaan Bluebird Pool BSD	1	29 Januari 2024	2 Februari 2024
2.	Pemahaman sistem dan aplikasi yang dipakai perusahaan dan observasi data dan proyek	2-3	5 Februari 2024	16 Februari 2024
3.	*Exploratory Data Analysis* (EDA)	4-6	19 Februari 2024	8 Maret 2024
	3.1. Data Exploration	4	19 Februari 2024	23 Februari 2024
	3.2. Data Cleansing	5	26 Februari 2024	1 Maret 2024
	3.3. Visualisasi Data	6	4 Maret 2024	8 Maret 2024
4.	Klasifikasi Penghasilan *Driver*	7-9	12 Maret 2024	28 Maret 2024

	4.1. *Import Library* SVM dan Pemilihan Fitur	7	12 Maret 2024	13 Maret 2024
	4.2. Pembagian Data	7	14 Maret 2024	15 Maret 2024
	4.3. Pemilihan Model	8	18 Maret 2024	22 Maret 2024
	4.4. Evaluasi Model	9	25 Maret 2024	28 Maret 2024
5.	*Clustering* Penghasilan *Driver*	10-12	1 April 2024	19 April 2024
	5.1. Import Library *Clustering* dan Menentukan Jumlah *Cluster* dengan *Elbow Method*	10	1 April 2024	5 April 2024
	5.2. Pelatihan Model K-Means	12	15 April 2024	17 April 2024
	5.3. Visualisasi Hasil *Clustering*	12	18 April 2024	19 April 2024
6.	Pembuatan *Dashboard* Penghasilan *Driver*	13-15	22 April 2024	10 Mei 2024
	6.1. Membuat *Worksheet* Visualisasi Data	13-14	22 April 2024	3 Mei 2024
	6.2. *Dashboard* Penghasilan *Driver*	15	6 Mei 2024	10 Mei 2024

(Sumber olahan peneliti, 2024)

Tabel 3.1 telah disusun untuk menggambarkan jadwal waktu dan detail pekerjaan yang dilakukan mahasiswa selama periode magang di Bluebird Pool BSD. Jadwal waktu dan rincian pekerjaan ini disusun berdasarkan aktivitas utama yang telah dilakukan oleh mahasiswa mulai dari Januari 2024 hingga Mei 2024.

3.2.1. *Onboarding* perusahaan Bluebird Pool BSD

Proses magang di Bluebird Pool BSD dimulai dengan mengikuti sesi presentasi pada tanggal 29 Januari 2024, yang memberikan gambaran umum tentang perusahaan. Presentasi ini memuat informasi mengenai struktur organisasi, peraturan perusahaan, dan sistem kerja yang dijelaskan oleh tim *Human Resource* (HR) Bluebird Pool BSD. Setelah presentasi, HR mengadakan tur untuk memperkenalkan berbagai departemen dan fasilitas di gedung perusahaan yang berlokasi di Jl. Cilenggang II No. 30, BSD, Cilenggang, Tangerang, Kota

Tangerang Selatan, Banten. HR secara rinci memperkenalkan berbagai ruangan di kantor dan rekan-rekan karyawan yang bekerja di departemen lain.

3.2.2. Pemahaman sistem dan aplikasi yang dipakai perusahaan dan observasi data dan proyek

Bluebird Pool BSD memilih Excel sebagai salah satu software yang digunakan karena kemudahan penggunaannya dan tampilan yang familiar di kalangan pengguna. Sebagai aplikasi Microsoft, Excel memungkinkan pengaturan data dalam sel-sel terstruktur dalam baris dan kolom. Data yang dimasukkan ke dalam lembar kerja dapat diolah dengan tingkat akurasi tinggi menggunakan berbagai rumus yang disediakan sehingga mempermudah pengguna. Selain itu, Excel juga memungkinkan pengguna untuk memvisualisasikan data yang telah diolah, termasuk pembuatan tabel, diagram, dan grafik. Akan tetapi, Excel memiliki keterbatasan dalam pemrosesan data yang kompleks, sehingga memerlukan *software* lain yang lebih canggih untuk melakukan pemrosesan data yang lebih lanjut.

Pengolahan data dilaksanakan menggunakan Jupyter Notebook dengan Python sebagai bahasa pemrograman utama karena sifatnya yang *open source*. Python telah umum digunakan dalam beragam tahapan analisis data dan evaluasi, memberikan dukungan yang diperlukan bagi *data analyst* untuk melakukan perbaikan. Python juga mendukung konsep *Object Oriented Programming* (OOP) dan dapat digunakan untuk membuat skrip serta *Application Programming Interface* (API) yang diperlukan. Dalam Jupyter Notebook, Python dapat digunakan dengan berbagai *library* yang dapat diimpor dan digunakan dalam proses penulisan kode. Pemilihan Jupyter Notebook didasarkan pada kenyamanan dan kebiasaan pribadi, sehingga menciptakan lingkungan kerja yang familiar dan sesuai dengan preferensi pengguna.

Selain penggunaan Excel, aplikasi Tableau juga digunakan untuk melakukan visualisasi data. Tableau dapat membuat berbagai jenis visualisasi, seperti grafik, peta, dan *dashboard* interaktif, yang memungkinkan pengguna untuk menggali

informasi yang berharga dari dataset dengan lebih baik. Visualisasi yang dihasilkan oleh Tableau membantu tim dalam memahami pola dan tren dalam data dengan lebih cepat dan efisien, serta mempermudah dalam menyajikan temuan analisis kepada pemangku kepentingan. Dengan demikian, penggunaan Tableau dalam proyek ini memberikan kontribusi yang signifikan dalam memahami dan mengkomunikasikan hasil analisis data.

3.2.3. Exploratory Data Analysis (EDA)

3.2.3.1. Data Exploration

Pembangunan visualisasi data menggunakan metode *exploratory data analysis* dimulai dengan tahapan *data exploration*. *Data exploration* dilakukan untuk mengetahui model visualisasi yang ingin dibangun serta informasi yang ingin diketahui.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Gambar 3. 1 *Import Library* EDA yang Dibutuhkan

Gambar 3. 1 menampilkan kode yang merupakan tahap awal dalam proses analisis data menggunakan bahasa pemrograman Python. Kode tersebut melibatkan pengimporan library yang diperlukan, termasuk pandas untuk manipulasi dan analisis data dalam bentuk tabel, numpy untuk operasi matematika yang melibatkan *array*, serta matplotlib.pyplot dan seaborn untuk visualisasi data.

```
In [5]: data = pd.read_excel("D:\hasilan_driver.xlsx")
        data.head(5)
```

Out[5]:

	Tgl	no_mobil	jenis	NIP	nama	gol	absen	SIO	keluar	masuk	update	hasil	fasilitas	range_hasil
0	1	BDE129	Transmover	291398	SUDRAJAT	CS	04:40:00	04:40:00	04:50:00	NaN	20:51:00	835620	STIKER	750-800
1	1	BDE131	Transmover	71338	ASEP	CM	18:11:00	18:11:00	18:37:00	NaN	11:06:00	812920	STIKER	750-800
2	1	BDE1340	Transmover	328408	TAUFIK	CS	07:52:00	07:52:00	08:08:00	NaN	23:45:00	472080	Non Stiker	400-500
3	1	BDE139	Transmover	313508	RINO SISWAHYUDI	CS	04:05:00	04:05:00	04:24:00	NaN	17:18:00	551880	STIKER	<750
4	1	BDE1411	Transmover	167272	MUJADI	CM	17:19:00	17:19:00	18:24:00	12:18:00	13:19:00	426392	STIKER	<750

Gambar 3. 2 Membaca Data Excel

Pada kode yang terdapat dalam gambar 3. 2 tersebut, data yang berupa file Excel dengan nama "hasilan_*driver*.xlsx" dibaca dan diubah menjadi

bentuk yang dapat digunakan oleh *software* Python menggunakan fungsi pd.read_excel() dari library Pandas. Fungsi ini memungkinkan pengguna untuk membaca file Excel dan mengkonversinya menjadi sebuah *dataframe* 'data'. Setelah data di baca, fungsi head(5) digunakan untuk menampilkan 5 baris pertama dari data tersebut.

```
In [7]: data.columns

Out[7]: Index(['Tgl', 'no_mobil', 'jenis', 'NIP', 'nama', 'gol', 'absen', 'SIO',
               'keluar', 'masuk', 'update', 'hasil', 'fasilitas', 'range_hasil'],
              dtype='object')
```

Gambar 3. 3 *Data Columns*

Gambar 3. 3 terdapat kode yang bertujuan untuk menampilkan kolom atau atribut apa saja yang terdapat pada data penghasilan *driver*. Pada data penghasilan *driver* Bluebird Pool BSD terdapat attribute Tgl, no_mobil, jenis, NIP, nama, gol, absen, SIO, keluar, masuk, update, hasil, fasilitas, dan range_hasil.

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 458 entries, 0 to 457
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Tgl          458 non-null    int64
 1   no_mobil     458 non-null    object
 2   jenis        458 non-null    object
 3   NIP          458 non-null    int64
 4   nama         458 non-null    object
 5   gol          458 non-null    object
 6   absen        458 non-null    object
 7   SIO          458 non-null    object
 8   keluar       455 non-null    object
 9   masuk        61 non-null     object
 10  update       458 non-null    object
 11  hasil        458 non-null    int64
 12  fasilitas    458 non-null    object
 13  range_hasil  458 non-null    object
dtypes: int64(3), object(11)
memory usage: 50.2+ KB
```

Gambar 3. 4 Informasi Data

Pada gambar 3. 4 bertujuan untuk melihat informasi tipe data yang ada pada data penghasilan *driver* Bluebird Pool BSD. Informasi ini sangat berguna untuk memahami keberadaan data, mengidentifikasi nilai yang hilang, dan memahami tipe data yang digunakan dalam *dataframe*.

3.2.3.2. Data Cleansing

Pada tahapan ini akan dilakukan proses *cleansing* terhadap data yang yang hilang atau adanya *missing values* pada data penghasilan *driver* Bluebird Pool BSD.

```
In [9]: data.isna().sum()

Out[9]: Tgl            0
        no_mobil       0
        jenis          0
        NIP            0
        nama           0
        gol            0
        absen          0
        SIO            0
        keluar         3
        masuk        397
        update         0
        hasil          0
        fasilitas      0
        range_hasil    0
        dtype: int64
```

Gambar 3. 5 Mengecek Missing Values

Pada gambar 3. 5, baris kode digunakan untuk mengidentifikasi nilai yang *null* dalam *dataframe* 'data'. Kode tersebut memberikan informasi yang bermanfaat dalam memahami seberapa banyak data yang kosong dalam *dataset*, yang nantinya dapat digunakan dalam proses pembersihan data atau menentukan langkah selanjutnya untuk mengatasi masalah tersebut. Dalam proses analisis data, melakukan *data cleansing* memiliki tujuan untuk menghindari adanya data yang tidak lengkap pada *dataset*.

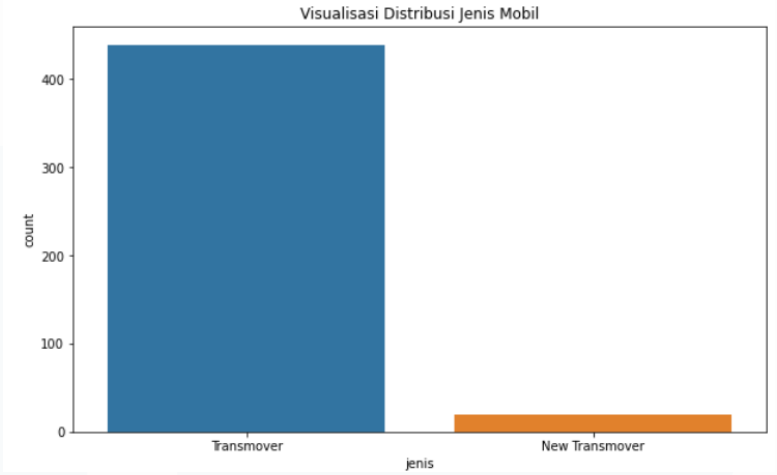
```
In [12]: data.fillna(0, inplace=True)
```

Gambar 3. 6 Mengisi Missing Value

Pada gambar 3. 6, baris kode digunakan untuk mengisi nilai kosong (NaN) dalam suatu *dataset* dengan nilai yang ditentukan. Dalam proses ini, nilai kosong akan digantikan dengan nilai nol sehingga *dataset* menjadi lebih stabil dan siap digunakan untuk analisis statistik atau visualisasi.

3.2.3.3. Visualisasi Data

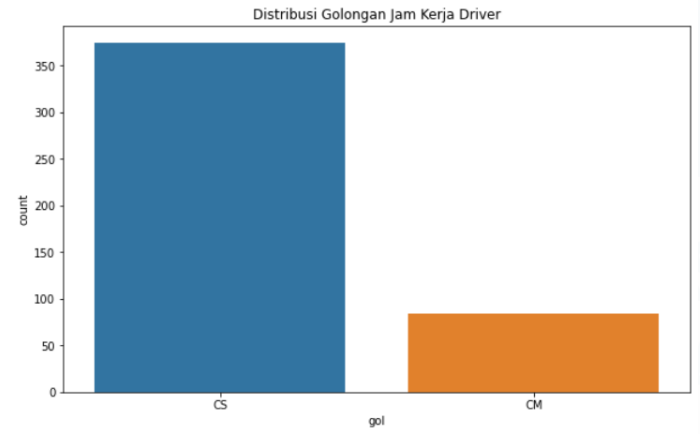
```
plt.figure(figsize=(10, 6))
sns.countplot(x='jenis', data=data)
plt.title('Visualisasi Distribusi Jenis Mobil')
plt.show()
```



Gambar 3. 7 Visualisasi Grafik Jenis Mobil

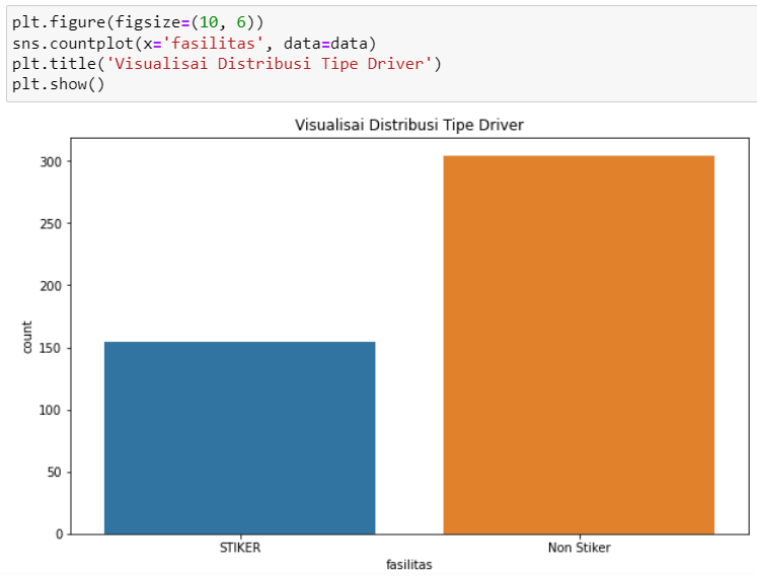
Gambar 3. 7 menampilkan kode dan hasil visualisasi berupa diagram batang yang menunjukkan data 'jenis' mobil yang digunakan oleh *driver* di Bluebird Pool BSD. Dari visualisasi tersebut, terlihat bahwa terdapat 2 jenis mobil yang digunakan. Analisis visualisasi tersebut menunjukkan perbedaan yang signifikan, di mana jenis Transmover menjadi yang paling banyak digunakan dibandingkan dengan jenis mobil lainnya.

```
plt.figure(figsize=(10, 6))
sns.countplot(x='gol', data=data)
plt.title('Distribusi Golongan Jam Kerja Driver')
plt.show()
```



Gambar 3. 8 Visualisasi Grafik Golongan *Driver*

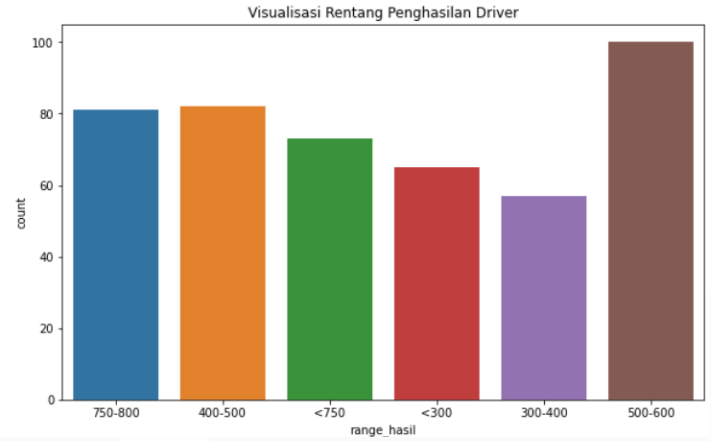
Di gambar 3. 8, terdapat kode yang berperan dalam pembuatan dan penayangan grafik batang yang menggambarkan distribusi dari kolom 'gol' dalam dataframe 'data'. Grafik tersebut menunjukkan bahwa golongan CS merupakan yang paling banyak diikuti oleh CM. Analisis grafik juga menampilkan perbedaan signifikan, di mana tipe Transmover memiliki jumlah yang lebih banyak dibandingkan dengan jenis mobil lainnya.



Gambar 3. 9 Visualisasi Grafik Tipe *Driver*

Di dalam gambar 3. 9, ada baris kode yang berfungsi untuk membuat dan menampilkan grafik batang yang menggambarkan distribusi dari kolom 'fasilitas' dalam *dataframe* 'data'. Hasil visualisasi tersebut menunjukkan bahwa ada 2 jenis mobil yang dipakai *driver* di Bluebird Pool BSD. Analisis grafik tersebut menunjukkan bahwa jumlah *driver* dengan fasilitas non-stiker jauh lebih banyak, mencapai sekitar 300 orang, dibandingkan dengan *driver* yang memiliki fasilitas stiker yang hanya sekitar 150 orang.

```
plt.figure(figsize=(10, 6))
sns.countplot(x='range_hasil', data=data)
plt.title('Visualisasi Rentang Penghasilan Driver')
plt.show()
```

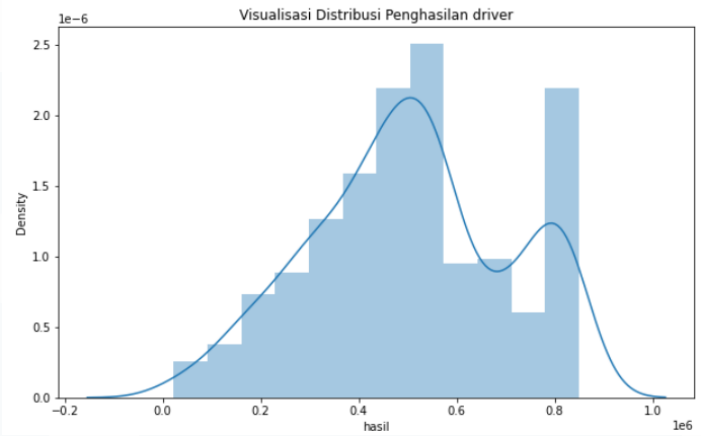


Gambar 3. 10 Visualisasi Rentang Penghasilan *Driver*

Pada gambar 3. 10, kode tersebut berperan dalam pembuatan serta menampilkan grafik batang yang memvisualisasikan distribusi dari kolom 'range_hasil' dalam *dataframe* 'data'. Grafik tersebut menampilkan distribusi penghasilan *driver* Bluebird dalam berbagai rentang. Analisis grafik menunjukkan bahwa rentang penghasilan terbesar berada di kisaran 500-600 ribu, diikuti oleh rentang 750-800 ribu dan 400-500 ribu yang juga memiliki jumlah *driver* yang signifikan. Rentang 300-400 ribu memiliki jumlah *driver* paling sedikit. Dengan demikian, hasil analisis menunjukkan bahwa sebagian besar *driver* berada dalam kategori penghasilan menengah, dengan jumlah yang lebih sedikit di kategori penghasilan rendah dan tinggi.

```
plt.figure(figsize=(10, 6))
sns.distplot(data['hasil'])
plt.title('Visualisasi Distribusi Penghasilan driver')
plt.show()
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Future
will be removed in a future version. Please adapt your code to use either `disp
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Gambar 3. 11 Visualisasi Histogram Rentang Penghasilan *Driver*

Gambar 3. 11 menunjukkan kode yang digunakan untuk membuat dan menampilkan grafik histogram yang menggambarkan distribusi jumlah kemunculan setiap nilai unik dalam kolom 'gol' dari dataframe 'data'. Histogram pada gambar tersebut menampilkan bahwa penghasilan *driver* yang miring ke kanan, menunjukkan bahwa mayoritas *driver* memiliki penghasilan yang berada di rentang yang lebih rendah, sementara ada beberapa *driver* dengan penghasilan yang jauh lebih tinggi.

3.2.4. Klasifikasi Penghasilan *Driver* Bluebird Pool BSD

Proyek ini dilakukan dengan tujuan untuk mengklasifikasikan penghasilan para *driver* Bluebird Pool BSD berdasarkan beberapa variabel. Data yang digunakan dalam proyek ini mencakup beberapa variabel kategorikal lainnya. Dalam proses klasifikasi, metode algoritma *Support Vector Machine* (SVM) merupakan salah satu metode *machine learning* yang efektif dalam melakukan klasifikasi. Dengan menggunakan SVM, proyek ini berusaha untuk mengidentifikasi pola atau tren dalam data yang dapat membantu dalam memprediksi penghasilan *driver* Bluebird Pool BSD berdasarkan variabel-

variabel yang ada. Proyek ini dilakukan dalam periode 13 Maret 2024 – 28 Maret 2024.

3.2.4.1. *Import Library* SVM dan Pemilihan Fitur

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.metrics import classification_report
```

Gambar 3. 12 Import Library SVM yang Dibutuhkan

Pada gambar 3. 12 menampilkan beberapa langkah yang umumnya dilakukan dalam proses pemodelan dengan memanfaatkan algoritma Support Vector Machine (SVM) untuk klasifikasi. Langkah pertama melibatkan import modul-modul yang diperlukan dari pustaka scikit-learn, seperti `train_test_split` digunakan untuk membagi data menjadi data *training* dan data *testing*, `StandardScaler` untuk penskalaan fitur, `OneHotEncoder` dan `LabelEncoder` untuk mengubah variabel kategorikal menjadi bentuk numerik, `ColumnTransformer` untuk menggabungkan transformasi pada kolom-kolom tertentu, `Pipeline` untuk menggabungkan beberapa langkah pemrosesan data, `SVC` sebagai model SVM, dan `classification_report` untuk mengevaluasi performa model. Dengan menggunakan pipeline, proses-proses ini dapat digabungkan secara bersamaan dan efisien.

```
label_encoder = LabelEncoder()
data['jenis'] = label_encoder.fit_transform(data['jenis'])
data['gol'] = label_encoder.fit_transform(data['gol'])
data['fasilitas'] = label_encoder.fit_transform(data['fasilitas'])
data['range_hasil'] = label_encoder.fit_transform(data['range_hasil'])
```

Gambar 3. 13 Encoding Variabel Kategorikal

Pada gambar 3. 13, kode digunakan untuk mengubah variabel kategorikal menjadi numerik menggunakan label_encoder. Proses ini penting untuk mempersiapkan data untuk pemodelan. Dalam proses ini, objek label_encoder dibuat menggunakan LabelEncoder(). Kemudian, setiap kolom variabel kategorikal dalam data diubah menjadi representasi numerik menggunakan

fit_transform(). Hasil transformasi disimpan kembali ke kolom yang sama dalam dataframe.

```
X = data[['gol', 'fasilitas', 'jenis']]
y = data['range_hasil']
```

Gambar 3. 14 Pemilihan Fitur dan Pelabelan Data

Pada gambar 3. 14, kode digunakan untuk membagi dataset menjadi dua bagian, satu bagian untuk fitur (X) dan satu bagian untuk target (y). Dalam proses ini, dipilih kolom 'gol', 'fasilitas', dan 'jenis' dari dataset sebagai fitur yang akan digunakan untuk melakukan prediksi pada bagian X. Sedangkan pada bagian y, dipilih kolom 'range_hasil' sebagai target yang akan diprediksi oleh model. Proses ini adalah langkah awal dalam mempersiapkan data untuk proses pelatihan model machine learning, di mana fitur dan target dipisahkan untuk keperluan analisis dan pelatihan model SVM.

3.2.4.2. Pembagian Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 3. 15 Pembagian Data

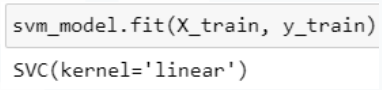
Di dalam gambar 3. 15, kode tersebut digunakan untuk membagi data menjadi empat bagian, yaitu X_train, X_test, y_train, dan y_test, dengan menggunakan fungsi train_test_split dari *library* scikit-learn. Dalam proses ini, data dibagi dengan proporsi 80:20, di mana 80% data digunakan untuk melatih model (X_train dan y_train) dan 20% digunakan untuk menguji model (X_test dan y_test). Penggunaan random_state=42 memastikan bahwa pembagian data dilakukan secara konsisten, sehingga memudahkan untuk mereproduksi hasil yang sama.

3.2.4.3. Pemilihan Model

```
from sklearn.svm import SVC
svm_model = SVC(kernel='linear', C=1.0)
```

Gambar 3. 16 Pemodelan SVM

Pada gambar 3. 16, sebuah model Support Vector Machine (SVM) untuk klasifikasi dibuat menggunakan *library* scikit-learn. SVM_model menggunakan *kernel* linear, yang berarti model ini akan mencoba untuk membuat garis linear yang memisahkan dua kelas data. Dalam proses ini, nilai C yang digunakan adalah 1.0, yang merupakan parameter regularisasi yang mengontrol penalti untuk kesalahan klasifikasi.



Gambar 3. 17 Pelatihan model SVM

Dalam gambar 3. 17, model Support Vector Machine (SVM) yang telah dibuat sebelumnya dilatih menggunakan data *training* (X_train dan y_train) dengan metode fit(). Proses pelatihan ini mengatur parameter model sedemikian rupa sehingga model dapat memahami pola yang ada dalam data latih dan dapat melakukan prediksi dengan akurat pada data uji. Proses pelatihan ini adalah langkah penting dalam pengembangan model machine learning karena akan menentukan kemampuan model dalam menggeneralisasi pola pada data baru.

3.2.4.4. Evaluasi Model

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Prediksi label menggunakan data testing
y_pred = svm_model.predict(X_test)

# Evaluasi model
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Classification Report: ")
print(classification_report(y_test, y_pred))

from sklearn.metrics import confusion_matrix
print("Confusion Matrix: ")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy:  0.41304347826086957
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        10
           1       0.00      0.00      0.00        18
           2       0.40      0.96      0.56        23
           3       0.56      0.28      0.37        18
           4       0.33      0.20      0.25        10
           5       0.41      0.69      0.51        13

    accuracy                           0.41        92
   macro avg       0.28      0.35      0.28        92
weighted avg       0.30      0.41      0.31        92

Confusion Matrix:
[[ 0  0  9  0  1  0]
 [ 0  0 16  0  2  0]
 [ 0  0 22  0  1  0]
 [ 0  0  0  5  0 13]
 [ 0  0  8  0  2  0]
 [ 0  0  0  4  0  9]]
```

Gambar 3. 18 Evaluasi Model

Kode pada gambar 3. 18 digunakan untuk mengevaluasi performa model SVM penghasilan *driver* per hari dengan menggunakan data *testing*. Prediksi label dilakukan dengan memanggil metode *predict* pada objek svm_model dan menyimpannya dalam variabel y_pred. Selanjutnya, berbagai metrik evaluasi dihitung dan ditampilkan, termasuk accuracy_score, classification_report, dan confusion_matrix dari *library* sklearn.metrics. Evaluasi hasil menunjukkan bahwa model memiliki akurasi sebesar 0.41 yang menandakan bahwa hanya 41% prediksi yang benar. Namun, analisis lebih mendalam melalui laporan klasifikasi menunjukkan variasi performa yang signifikan di antara berbagai kelas. Kelas 1 mencakup penghasilan driver kurang dari 300 ribu, kelas 2 mencakup penghasilan antara 300 hingga 400 ribu, kelas 3 mencakup penghasilan antara 400 hingga 500 ribu, kelas 4 mencakup penghasilan antara 500 hingga 600 ribu, kelas 5 mencakup penghasilan hingga 750 ribu, dan kelas 6 mencakup penghasilan antara 750 hingga 800 ribu. Pada kelas 0 dan 1, *precision* dan *recall* adalah nol, yang menunjukkan bahwa model tidak mampu mengenali *instance* dari kelas-kelas tersebut dengan benar. Sedangkan untuk

kelas 2, model menunjukkan performa yang sangat baik dengan *precision* sempurna sebesar 1.00 dan *recall* hampir sempurna sebesar 0.96, mengindikasikan bahwa hampir semua *instance* kelas 2 diidentifikasi dengan benar oleh model. Pada kelas 3 dan 4, *precision* dan *recall* berada pada level yang cukup rendah, masing-masing di bawah 0.40, yang menunjukkan bahwa model sering salah dalam memprediksi *instance* dari kelas-kelas ini. Kelas 5 menunjukkan performa yang lebih baik dibandingkan kelas 3 dan 4, dengan *precision* sebesar 0.41 dan *recall* sebesar 0.69. *Confusion matrix* yang ditampilkan juga menunjukkan pola kesalahan prediksi, di mana banyak *instance* dari kelas 0 dan 1 salah diklasifikasikan sebagai kelas lain, terutama kelas 2. Kesimpulannya, model SVM ini bekerja dengan baik untuk beberapa kelas tertentu seperti kelas 2 dan 5 meskipun terdapat masalah dalam mengenali *instance* dari kelas 0 dan 1 sehingga menunjukkan perlunya perbaikan model atau penyesuaian data untuk mencapai performa yang lebih seimbang.

3.2.5. *Clustering Driver* Bluebird Berdasarkan Penghasilan

Proyek clustering ini dilakukan dengan tujuan untuk mengelompokkan para *driver* Bluebird berdasarkan tingkat penghasilan mereka. Dalam proyek ini, metode clustering digunakan untuk mengidentifikasi pola-pola dalam data penghasilan *driver*. Salah satu teknik *unsupervised learning* yang dapat membantu dalam mengelompokkan data ke dalam kelompok-kelompok yang memiliki karakteristik serupa tanpa adanya label kelas adalah algoritma *clustering*. Proyek ini dilakukan dalam periode 1 April 2024 – 19 April 2024.

3.2.5.1. *Import Library Clustering* dan Menentukan Jumlah *Cluster* dengan *Elbow Method*

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

Gambar 3. 19 *Import Library Clustering*

Pada gambar 3. 19, kode teserbut berfungsi untuk mengimpor dua *library* dari sklearn, yaitu KMeans yang digunakan untuk melakukan *clustering* atau pengelompokan data berdasarkan kesamaan fitur dan

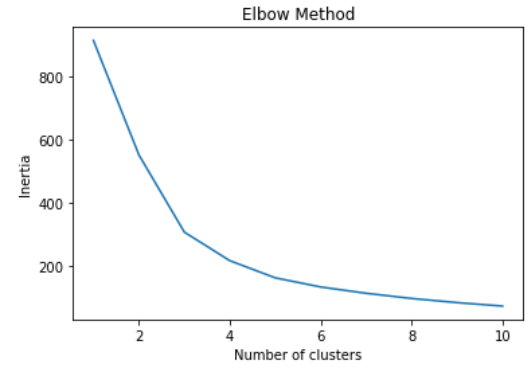
StandardScaler yang berfungsi untuk menstandarkan fitur-fitur dalam dataset dengan menghilangkan *mean* dan menskalakan ke varians unit. Dalam proses ini, dilakukan untuk memastikan bahwa setiap fitur memiliki skala yang sama, sehingga algoritma *clustering* dapat bekerja dengan lebih efektif.

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Menghitung inertia untuk jumlah cluster 1 hingga 10
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(data_scaled)
    inertia.append(kmeans.inertia_)

# Menampilkan Elbow Method
plt.plot(range(1, 11), inertia)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py
n Windows with MKL, when there are less chunks than available thread
_NUM_THREADS=2.
  warnings.warn(
```



Gambar 3. 20 Menentukan Jumlah *Cluster*

Pada gambar 3. 20, inertia untuk jumlah *cluster* dari 1 hingga 10 dihitung. Inertia adalah ukuran seberapa baik data dikelompokkan oleh algoritma KMeans, di mana nilai yang lebih rendah menunjukkan pengelompokan yang lebih baik. Dalam proses ini, objek KMeans dibuat dengan jumlah *cluster* sesuai nilai tersebut, menggunakan inisialisasi `k-means++` dan `random_state=42` untuk memastikan hasil yang konsisten. Model KMeans dilatih menggunakan data yang telah diskalakan (`data_scaled`), dan nilai inertia dari model tersebut ditambahkan ke dalam daftar `inertia`. Setelah itu, metode Elbow ditampilkan

dengan memplot jumlah *cluster* melawan nilai inertia. Grafik ini digunakan untuk menentukan jumlah *cluster* yang optimal dengan melihat titik di mana penurunan inertia mulai melambat secara signifikan.

3.2.5.2. Pelatihan Model K-Means

```
n_clusters = 3

kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
kmeans.fit(data_scaled)
```

Gambar 3. 21 Memilih Jumlah *Cluster* dan Melatih Model K-Means

Kode pada gambar 3. 21 memilih jumlah *cluster* berdasarkan hasil dari metode Elbow yang ditentukan menjadi 3 *cluster*. Setelah jumlah cluster ditetapkan, model K-Means dilatih menggunakan data yang telah diskalakan. Dalam proses ini, objek KMeans dibuat dengan parameter n_clusters=3, inisialisasi k-means++, dan random_state=42 untuk konsistensi hasil. Model tersebut kemudian dilatih (*fit*) menggunakan data yang telah diskalakan (data_scaled).

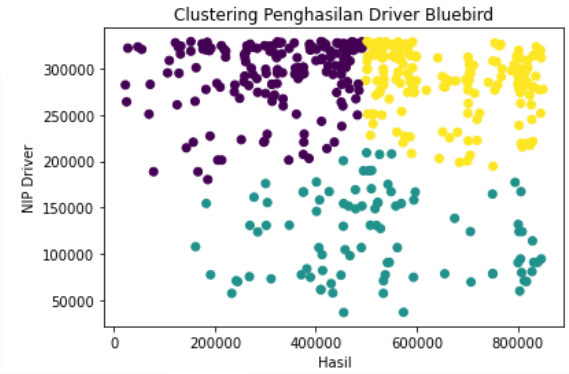
```
data['cluster'] = kmeans.labels_
```

Gambar 3. 22 Menambahkan Label *Cluster* ke *Dataset*

Pada gambar 3. 22, hasil pengelompokan dari model K-Means ditambahkan ke dalam dataset. Dalam proses ini, kolom baru bernama 'cluster' dibuat dalam dataset `data`, dan label dari model K-Means (`kmeans.labels_`) yang menunjukkan cluster mana setiap data termasuk, disimpan dalam kolom tersebut. Dengan demikian, setiap data pada dataset kini memiliki informasi mengenai cluster yang dihasilkan dari proses clustering.

3.2.5.3. Visualisasi Hasil *Clustering*

```
plt.scatter(data['hasil'], data['NIP'], c=data['cluster'], cmap='viridis')
plt.title('Clustering Penghasilan Driver Bluebird')
plt.xlabel('Hasil')
plt.ylabel('NIP Driver')
plt.show()
```



Gambar 3. 23 Visualisasi Hasil Clustering

Kode pada gambar 3. 23 digunakan untuk membuat visualisasi data pengelompokan (*clustering*) penghasilan *driver* Bluebird dengan menggunakan pustaka Matplotlib dalam Python. Dalam visualisasi ini, sumbu x mewakili penghasilan *driver* (`hasil`), sementara sumbu y mewakili NIP *driver* (`NIP`). Hasil *clustering* menunjukkan bahwa *driver* Bluebird dapat dikelompokkan menjadi 3 *cluster* berdasarkan penghasilan mereka. Dalam visualisasi, *cluster* pertama yang berwarna ungu memiliki penghasilan rata-rata yang relatif rendah, *cluster* kedua yang berwarna biru memiliki penghasilan rata-rata yang relatif menengah, dan *cluster* ketiga yang berwarna kuning memiliki penghasilan rata-rata yang relatif sangat tinggi.

3.2.6. Pembuatan *Dashboard* Penghasilan *Driver*

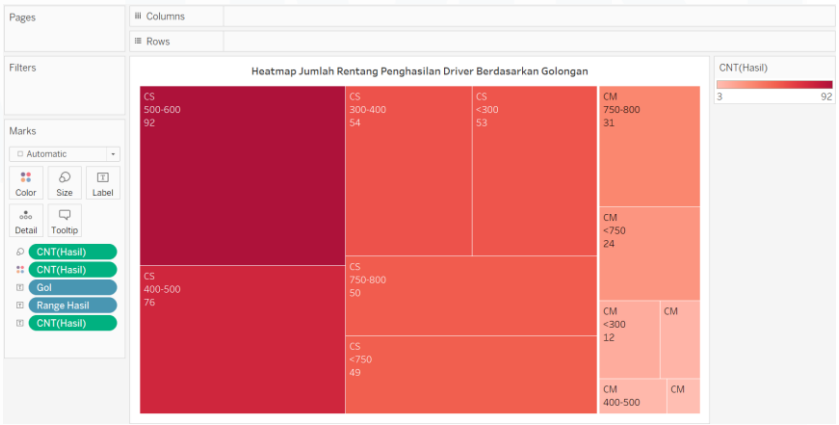
Pembuatan *dashboard* penghasilan driver dilaksanakan mulai tanggal 22 April hingga 10 Mei. Dalam proses ini, *tools* Tableau dipakai untuk menghasilkan *dashboard*. Data yang dipergunakan adalah data penghasilan driver. Dalam pembuatan *dashboard*, pemilihan jenis *chart* bergantung pada tipe data yang akan divisualisasikan karena tidak ada spesifikasi yang diberikan oleh pengguna terkait jenis *chart* yang harus digunakan.

3.2.6.1. Membuat *Worksheet* Visualisasi Data

Berikut ini adalah tampilan dari setiap lembar kerja (*worksheet*) yang akan disatukan menjadi satu dalam *dashboard*.

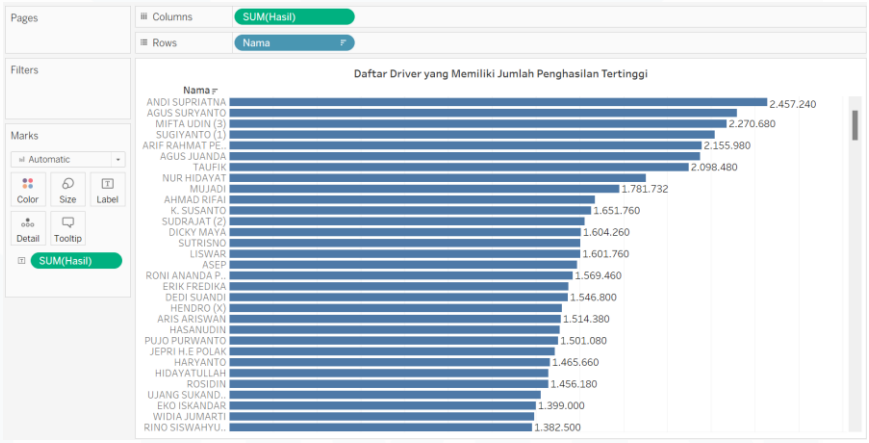
Gambar 3. 24 *Worksheet* Rata-Rata Penghasilan *Driver* Berdasarkan Jenis Mobil

Pada gambar 3.24, *worksheet* visualisasi menampilkan perbandingan rata-rata penghasilan *driver* berdasarkan jenis mobil yang digunakan. Dalam visualisasi ini, terdapat dua jenis mobil yang dibandingkan, yaitu New Transmover dan Transmover. Dari grafik batang tersebut, terlihat bahwa rata-rata penghasilan *driver* yang menggunakan Transmover lebih tinggi dibandingkan dengan rata-rata penghasilan *driver* yang menggunakan New Transmover.



Gambar 3. 25 *Worksheet* Jumlah Rentang Penghasilan *Driver* Berdasarkan Golongan

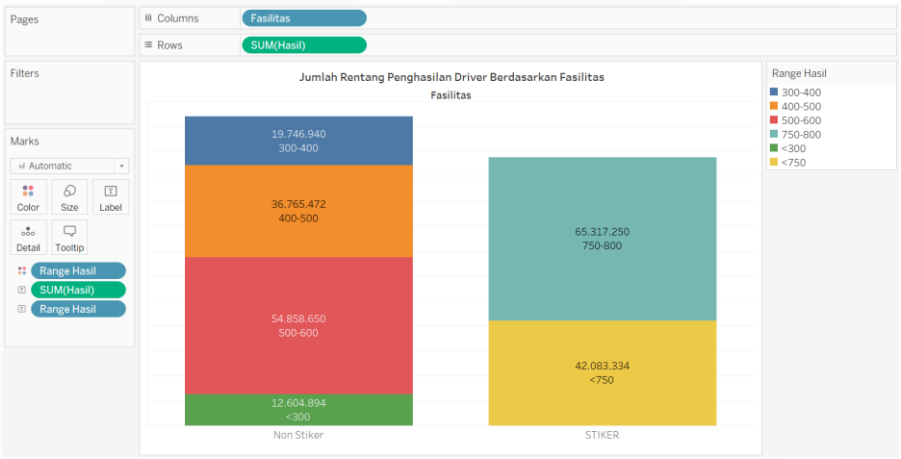
Pada gambar 3.25, visualisasi menunjukkan heatmap jumlah rentang penghasilan *driver* berdasarkan golongan. Dalam heatmap ini, warna yang lebih gelap menunjukkan jumlah yang lebih tinggi, sementara warna yang lebih terang menunjukkan jumlah yang lebih rendah. Hasil heatmap menunjukkan bahwa golongan "CS" dengan rentang penghasilan 500-600 memiliki jumlah terbanyak, yaitu 92 *driver*. Sementara itu, golongan "CS" dengan rentang penghasilan 300-400 dan kurang dari 300 memiliki jumlah yang cukup signifikan, masing-masing 54 dan 53 *driver*. Rentang penghasilan lainnya, seperti "CS" 400-500, "CS" 750-800, dan "CS" kurang dari 750, serta beberapa golongan "CM," memiliki jumlah yang lebih bervariasi dengan angka yang lebih rendah. Heatmap ini memberikan gambaran yang jelas mengenai distribusi jumlah *driver* berdasarkan rentang penghasilan dan golongan mereka.



Gambar 3. 26 *Worksheet* Daftar *Driver* yang Memiliki Jumlah Penghasilan Tertinggi

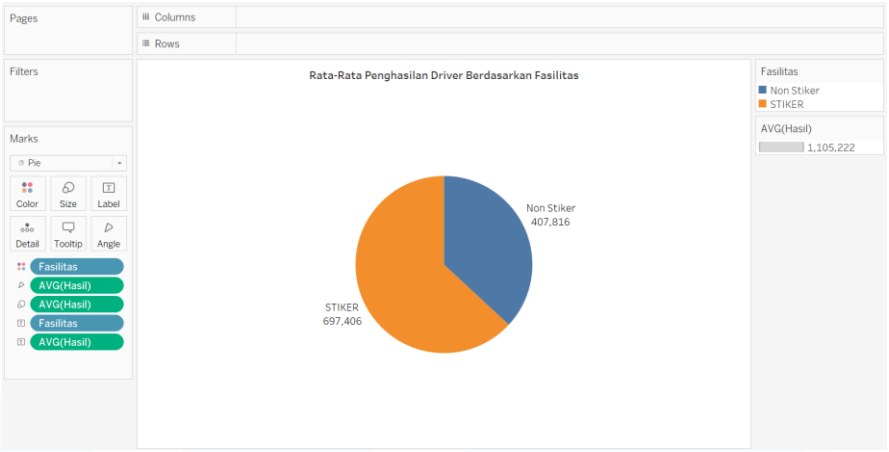
Pada gambar 3.26, visualisasi menampilkan daftar *driver* dengan jumlah penghasilan tertinggi. Dalam grafik batang horizontal ini, nama-nama *driver* ditampilkan pada sumbu vertikal, sedangkan jumlah penghasilan ditampilkan pada sumbu horizontal. Batang yang lebih panjang menunjukkan penghasilan yang lebih tinggi, dan urutan *driver* diatur berdasarkan total penghasilan dari yang tertinggi ke terendah. Visualisasi

ini memudahkan untuk melihat perbandingan penghasilan di antara *driver* dan mengidentifikasi siapa saja yang memiliki penghasilan tertinggi.



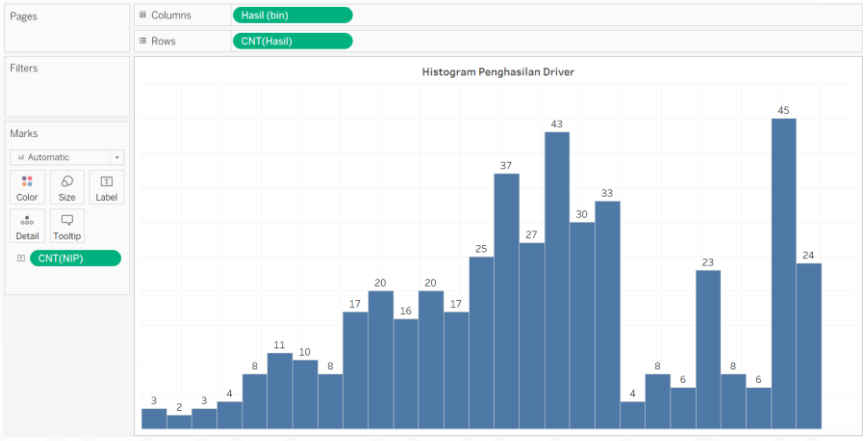
Gambar 3. 27 *Worksheet* Jumlah Rentang Penghasilan *Driver* Berdasarkan Fasilitas

Visualisasi pada gambar 3. 27 menunjukkan jumlah rentang penghasilan *driver* berdasarkan fasilitas, yang dibedakan menjadi dua kategori: Non Stiker dan Stiker. Dalam kategori Non Stiker, rentang penghasilan terbesar berada pada rentang 500-600 dengan jumlah 54.858.650, diikuti oleh rentang 400-500 sebesar 36.765.472, rentang 300-400 sebesar 19.746.940, dan rentang di bawah 300 sebesar 12.604.894. Sementara itu, pada kategori Stiker, rentang penghasilan terbesar berada pada rentang 750-800 dengan jumlah 65.317.250, diikuti oleh rentang di bawah 750 sebesar 42.083.334. Dalam visualisasi, rentang penghasilan dalam kategori Non Stiker lebih bervariasi dibandingkan dengan kategori Stiker.



Gambar 3. 28 *Worksheet* Rata-Rata Penghasilan *Driver* Berdasarkan Fasilitas

Visualisasi pada gambar 3. 28 menampilkan perbandingan rata-rata penghasilan *driver* berdasarkan fasilitas. Dalam visualisasi ini, terdapat dua jenis fasilitas yang dibandingkan, yaitu stiker dan non stiker. Dari pie chart tersebut, terlihat bahwa rata-rata penghasilan *driver* yang memiliki fasilitas stiker lebih tinggi dibandingkan dengan rata-rata penghasilan *driver* yang memiliki fasilitas non-stiker.



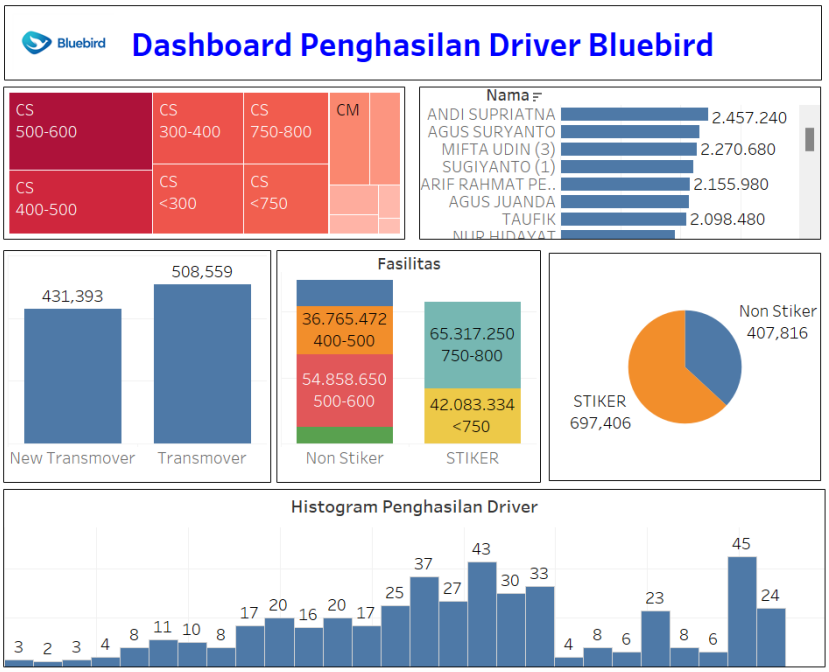
Gambar 3. 29 *Worksheet* Histogram Penghasilan *Driver*

Histogram yang terdapat pada gambar 3.29 menggambarkan distribusi penghasilan *driver*. Data menunjukkan jumlah *driver* di setiap rentang penghasilan. Dalam histogram, rentang penghasilan yang paling banyak ditempati adalah pada kisaran tinggi, yaitu dengan 45 *driver*, diikuti

oleh rentang yang sedikit lebih rendah dengan 43 *driver*. Rentang penghasilan lainnya memiliki variasi yang lebih kecil, dengan jumlah *driver* berkisar antara 2 hingga 37. Puncak kedua tertinggi berada pada rentang penghasilan dengan 37 *driver*. Dari histogram ini dapat dilihat bahwa ada kecenderungan lebih banyak *driver* dengan penghasilan pada dua rentang tertentu, menunjukkan ketimpangan dalam distribusi penghasilan *driver*.

3.2.6.2. *Dashboard* Penghasilan *Driver*

Beberapa contoh lembar kerja (*worksheet*) dari *dashboard* penghasilan *driver*, yaitu Gambar 3.24 hingga 3.29, akan disatukan menjadi satu bagian pada *dashboard* penghasilan *driver*. Gabungan dari Gambar 3.24 hingga 3.29 dapat dilihat sebagai berikut.



Gambar 3. 30 *Dashboard* Penghasilan *Driver* Bluebird

Dashboard pada gambar 3.30 menggambarkan berbagai aspek penghasilan *driver* Bluebird. Pada bagian kiri atas, visualisasi berbentuk treemap menunjukkan rentang penghasilan *driver* dalam kategori CS dan CM. Di bagian kanan atas, terdapat daftar nama *driver* dengan penghasilan

tertinggi, di mana Andi Supriatna memiliki penghasilan tertinggi sebesar 2.457.240.

Grafik batang di bagian tengah menunjukkan jumlah penghasilan berdasarkan tipe kendaraan, dengan Transmover menghasilkan lebih banyak dibandingkan New Transmover. Visualisasi fasilitas memperlihatkan bahwa penghasilan terbesar berada di kategori Stiker dengan rentang 750-800 sebesar 65.317.250, sedangkan pada kategori Non Stiker penghasilan tertinggi berada pada rentang 500-600 sebesar 54.858.650. Diagram lingkaran menunjukkan distribusi jumlah *driver* dengan fasilitas Non Stiker dan Stiker, di mana lebih banyak *driver* menggunakan fasilitas Stiker (697.406) dibandingkan Non Stiker (407.816).

Terakhir, histogram di bagian bawah menggambarkan distribusi penghasilan *driver*, dengan puncak pada rentang tertinggi dengan 45 *driver* dan rentang kedua tertinggi dengan 43 *driver*, menunjukkan konsentrasi penghasilan pada rentang tertentu. Secara keseluruhan, *dashboard* ini memberikan gambaran komprehensif tentang berbagai faktor yang mempengaruhi penghasilan *driver* Bluebird, seperti tipe kendaraan, fasilitas yang digunakan, dan distribusi penghasilan di antara para *driver*.

3.3 Kendala yang Ditemukan

Secara keseluruhan, proses pelaksanaan program magang di Bluebird Pool BSD telah berjalan lancar, namun selama menjalani program magang, beberapa kendala muncul yang menghambat waktu penyelesaian tugas atau pekerjaan.

a. Waktu yang terbatas untuk bertemu dengan *supervisor* untuk berdiskusi mengenai tugas dikarenakan *supervisor* memiliki banyak kegiatan dan keperluan dikantor maupun diluar kantor.
b. Tidak ada penyediaan komputer dari Bluebird Pool BSD sehingga penggunaan laptop pribadi yang sudah lambat untuk menganalisis data cukup menghambat pekerjaan.

c. Rasa minder dan ketidaknyamanan dalam beradaptasi dengan situasi lingkungan kantor juga menjadi kendala, terutama pada awal magang ketika harus berinteraksi dengan tim dan atasan yang baru.

3.4 Solusi atas Kendala yang Ditemukan

a. Zoom *meeting* digunakan untuk berdiskusi dan membahas tugas dengan supervisor.
b. Kendala minimnya komputer membuat waktu di luar jam magang dimanfaatkan untuk menganalisis data sehingga *progress* pekerjaan dapat dipaparkan keesokan harinya di kantor.
c. Komunikasi yang efektif dengan rekan kerja dan atasan baru dilakukan untuk memperoleh pemahaman yang lebih baik tentang lingkungan kerja serta tugas yang harus dilakukan, menjaga etika kerja yang baik, dan menunjukkan sikap profesional dalam setiap interaksi dengan tim dan atasan untuk membangun reputasi yang positif.