Data Mining dan Visualisasi C

FINAL PROJECT

BIKE BUYERS DATASET ANALYSIS

Dosen Pengampu:
Irhamah, S.Si, M.Si., Ph.D.

AYUNDA FATIKHA
5003221023

OVERVIEW

01 Dataset Problem Statement

02 *Pre-Processing*

03 *Summary Statistics and Visualization*

04 Feature Selection / Extraction

05 Classification

06 Training-Testing

07 Perbandingan dan Pemilihan Metode Terbaik

08 Kesimpulan

DATASET
PROBLEM STATEMENT

Dalam beberapa tahun terakhir, minat masyarakat terhadap sepeda semakin meningkat. Hal ini didorong oleh berbagai faktor, seperti kesadaran akan gaya hidup sehat, kemacetan lalu lintas, dan upaya mengurangi emisi karbon. Meningkatnya minat ini berdampak pada industri sepeda, baik dari segi produksi maupun penjualan.

DIharapkan nantinya dengan analisis ini, dapat mengetahui karakteristik serta faktor-faktor yang memengaruhi pelanggan melakukan pembelian sepeda. Serta dapat membuat prediksi untuk memperkirakan pembelian sepeda.

Sumber Data

Dataset mengenai ‘Bike Buyers’ ini berasal dari Kaggle dimana digunakan untuk memprediksi kemungkinan seseorang membeli sepeda berdasarkan beberapa faktor prediksi. Adapun variabel yang ada pada dataset ‘Bike Buyers’ sebagai berikut.

ID
Marital Status
Gender
Income
Children
Education
Occupation
Home Owner
Cars
Commute Distance
Region
Age
Purchased Bike

Sumber Data

PRE-PROCESSING

Define Dataset

```
bike = pd.read_csv('bikebuyers.csv', sep=";")
```

```
bike.head()
```

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	12496	Married	Female	40000.0	1.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	Europe	42.0	0
1	24107	Married	Male	30000.0	3.0	Partial College	Clerical	Yes	1.0	0-1 Miles	Europe	43.0	0
2	14177	Married	Male	80000.0	5.0	Partial College	Professional	No	2.0	2-5 Miles	Europe	60.0	0
3	24381	Single	NaN	70000.0	0.0	Bachelors	Professional	Yes	1.0	5-10 Miles	Pacific	41.0	1
4	25597	Single	Male	30000.0	0.0	Bachelors	Clerical	No	0.0	0-1 Miles	Europe	36.0	1

```
bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                1000 non-null   int64
 1   Marital Status    993 non-null    object
 2   Gender            989 non-null    object
 3   Income            994 non-null    float64
 4   Children          992 non-null    float64
 5   Education         1000 non-null   object
 6   Occupation        1000 non-null   object
 7   Home Owner        996 non-null    object
 8   Cars              991 non-null    float64
 9   Commute Distance  1000 non-null   object
 10  Region            1000 non-null   object
 11  Age               992 non-null    float64
 12  Purchased Bike    1000 non-null   int64
```

Diketahui bahwa data sudah sesuai dengan kriteria yang ditentukan yaitu memuat minimal 10 variabel.

Drop Kolom “ID”

Kolom “ID” tidak diperlukan pada analisis.

```
bike = bike.drop(['ID'], axis = 1)
bike.head()
```

	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	Married	Female	40000.0	1.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	Europe	42.0	0
1	Married	Male	30000.0	3.0	Partial College	Clerical	Yes	1.0	0-1 Miles	Europe	43.0	0
2	Married	Male	80000.0	5.0	Partial College	Professional	No	2.0	2-5 Miles	Europe	60.0	0
3	Single	NaN	70000.0	0.0	Bachelors	Professional	Yes	1.0	5-10 Miles	Pacific	41.0	1
4	Single	Male	30000.0	0.0	Bachelors	Clerical	No	0.0	0-1 Miles	Europe	36.0	1

Cek Missing Value

```
bike.isnull().sum()/bike.shape[0]*100
```

	0
Marital Status	0.7
Gender	1.1
Income	0.6
Children	0.8
Education	0.0
Occupation	0.0
Home Owner	0.4
Cars	0.9
Commute Distance	0.0
Region	0.0
Age	0.8
Purchased Bike	0.0

Karena nilai missing value pada tiap variabel tidak terlalu besar, dilakukan **imputasi data** sebagai solusi untuk mengatasi missing value.

Imputasi Pada Missing Value

```python
# Mengisi missing value untuk variabel kategorikal (object)
categorical_columns = ['Marital Status', 'Gender', 'Home Owner', 'Education', 'Occupation', 'Commute Distance', 'Region']
for col in categorical_columns:
    bike[col].fillna(bike[col].mode()[0], inplace=True)

# Mengisi missing value untuk variabel numerik (float64)
numerical_columns = ['Income', 'Children', 'Cars', 'Age']
for col in numerical_columns:
    bike[col].fillna(bike[col].median(), inplace=True)
```

Cek Missing Value setelah Imputasi

```python
print(bike.isnull().sum())
```

```
Marital Status      0
Gender              0
Income              0
Children            0
Education           0
Occupation          0
Home Owner          0
Cars                0
Commute Distance    0
Region              0
Age                 0
Purchased Bike      0
```

Data sudah tidak terdapat missing value.

Cek Inconsistent Data

```python
#inconsistent in gender column
bike.groupby(['Gender']).size()
```

Gender	0
Female	489
Male	511

```python
#inconsistent in maritial status column
bike.groupby(['Marital Status']).size()
```

Marital Status	0
Married	542
Single	458

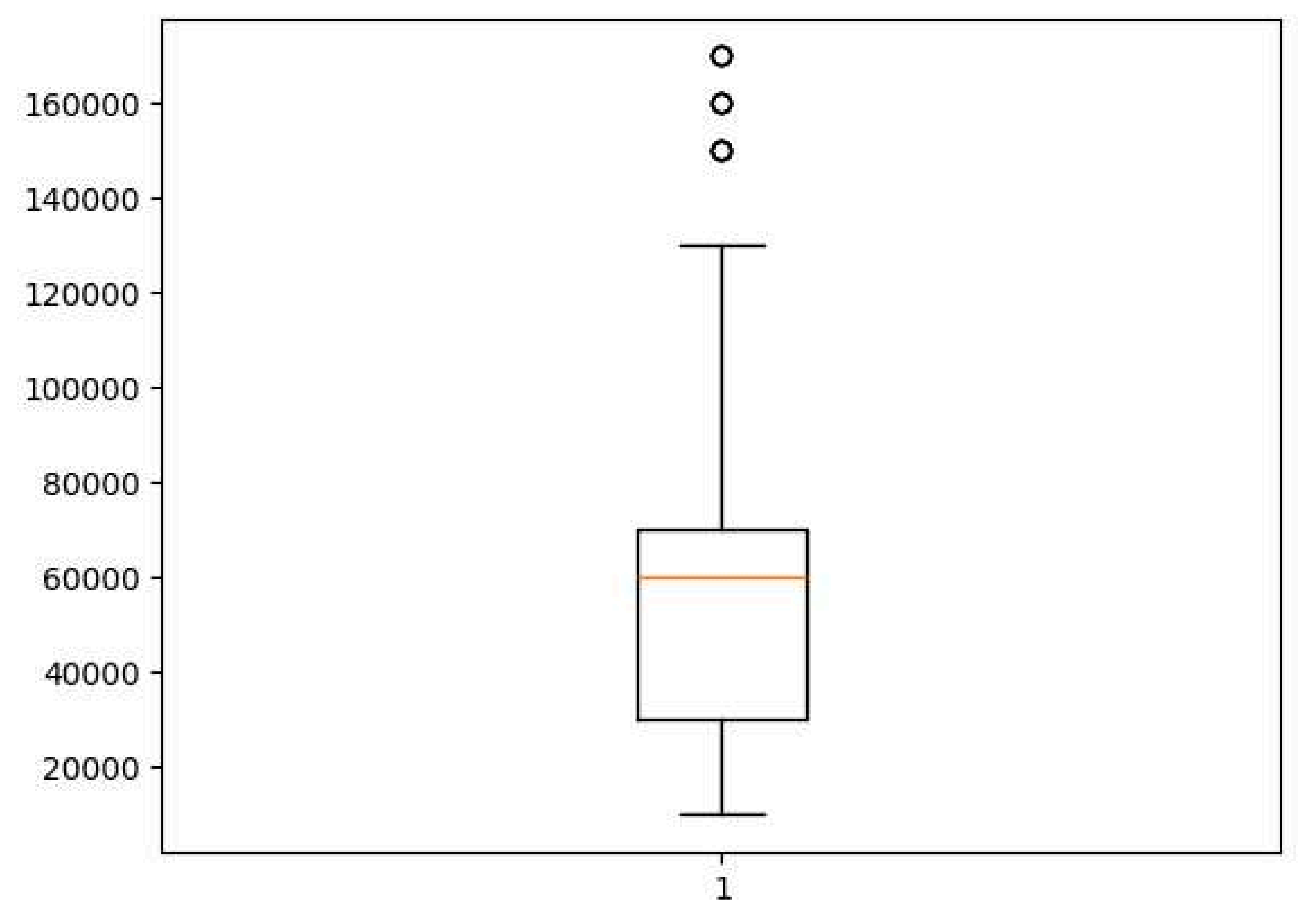
```python
#inconsistent in region column
bike.groupby(['Region']).size()
```

Region	0
Europe	300
North America	508
Pacific	192

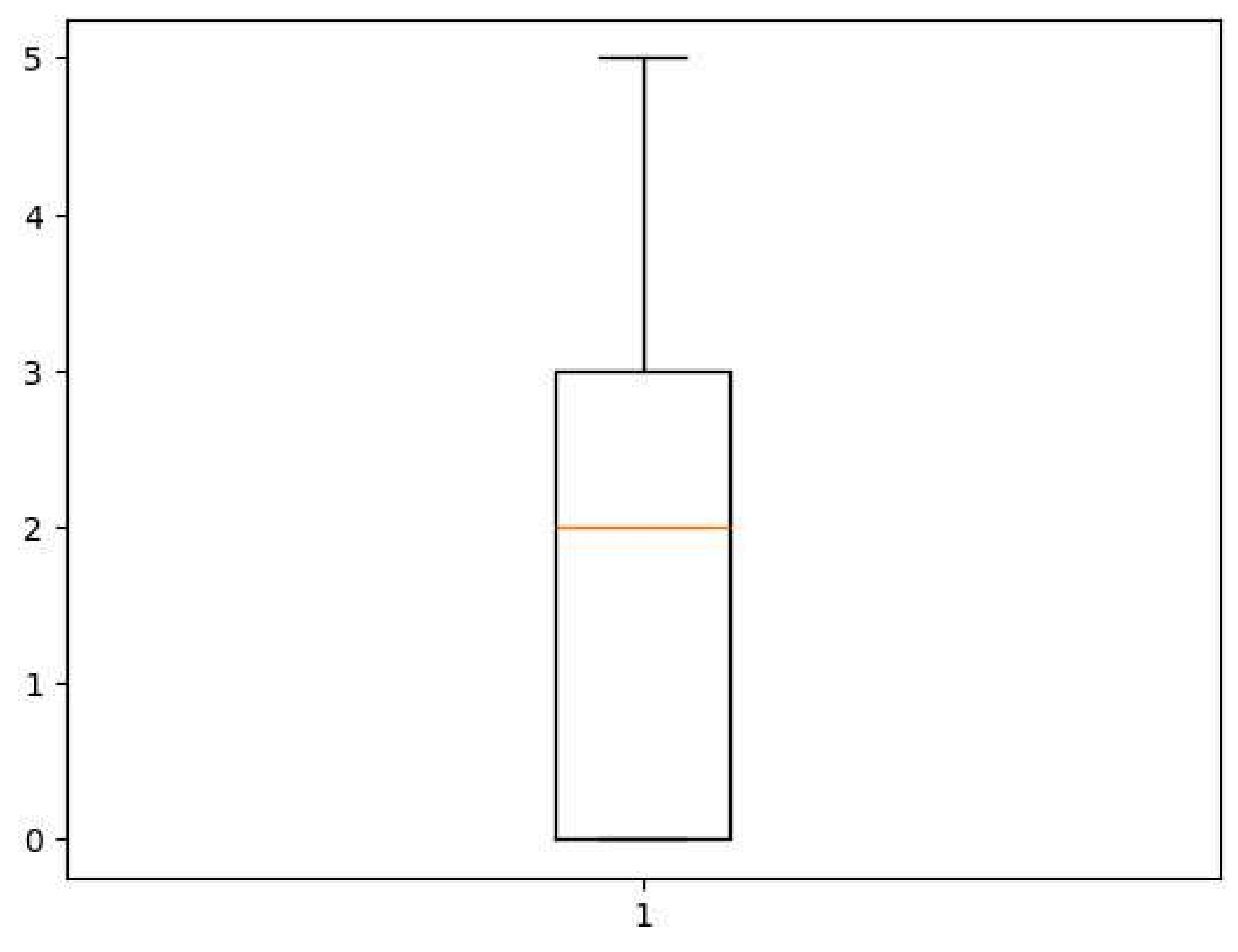
Data tidak menunjukkan inconsistent di tiap kategorinya.

Pengecekan Outliers

```
plt.boxplot(bike['Income'])
```




```
plt.boxplot(bike['Children'])
```



Terdapat outlier pada data ‘Income’, sedangkan pada data ‘Children’ tidak terindikasi adanya outliers.

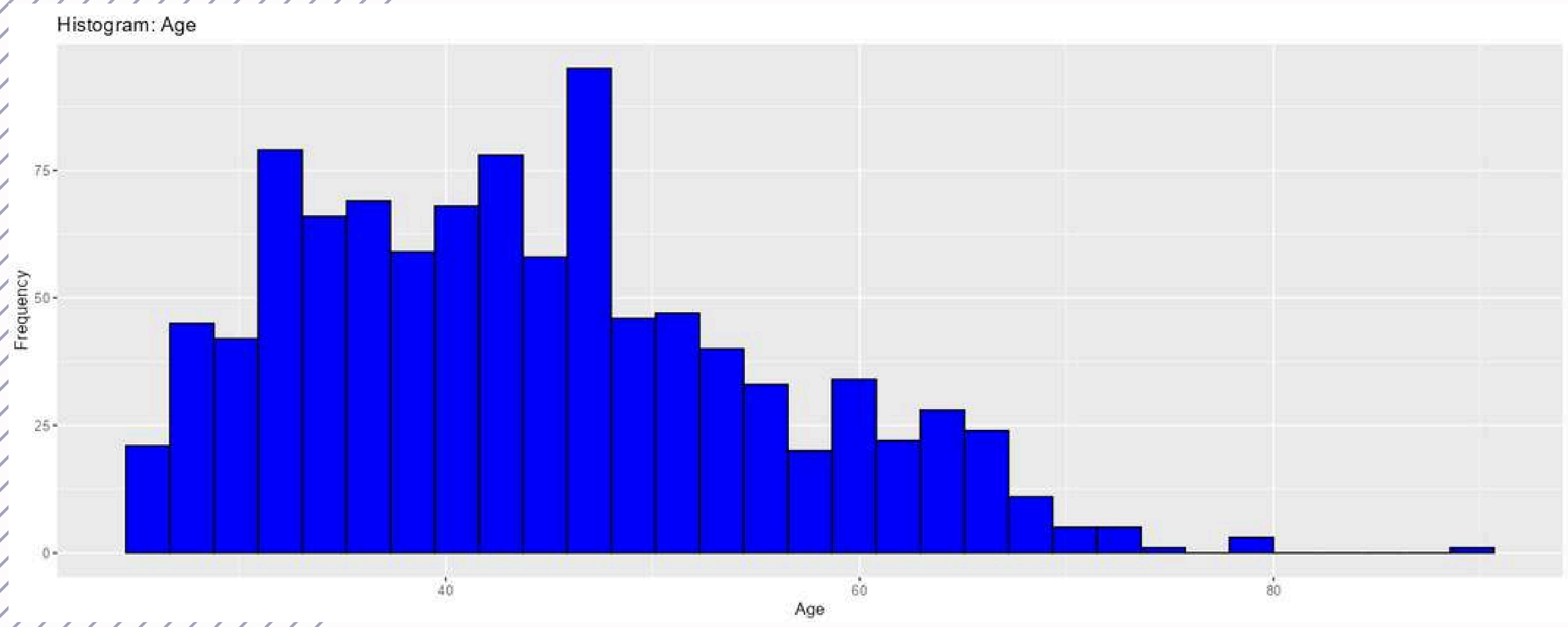
SUMMARY STATISTICS AND VISUALIZATION

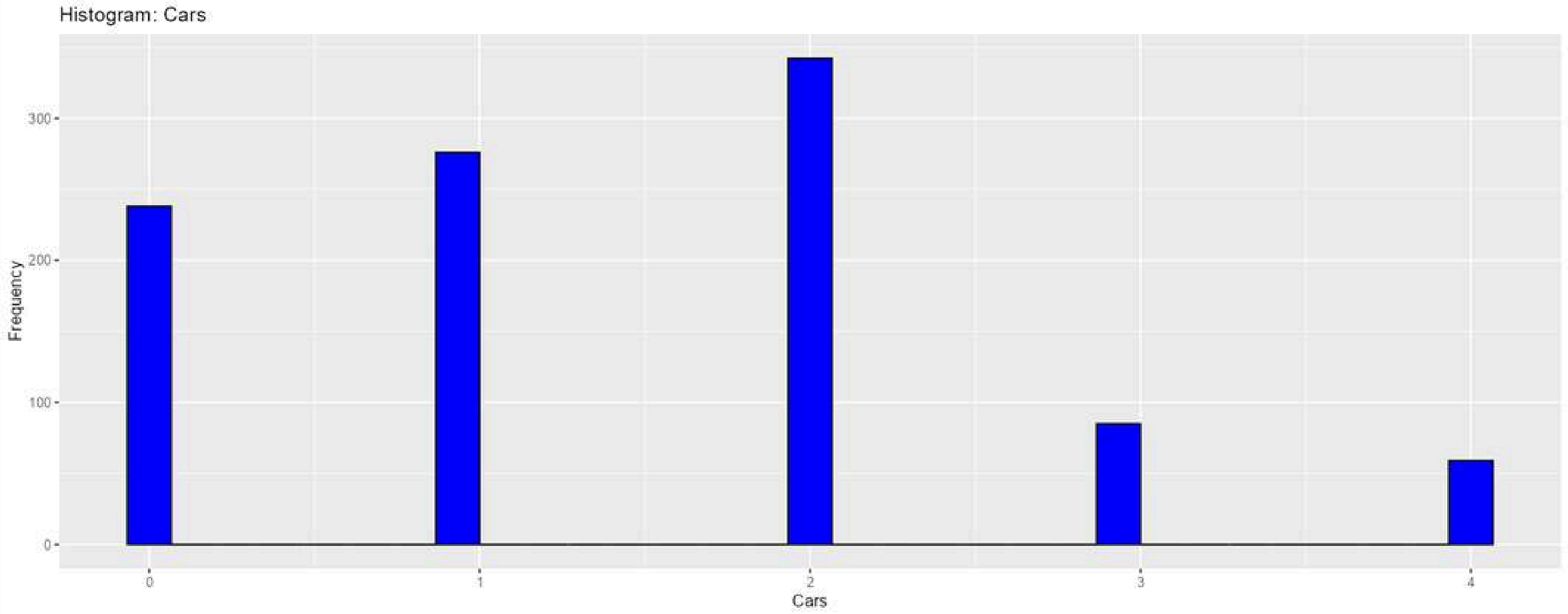
Summary Statistics

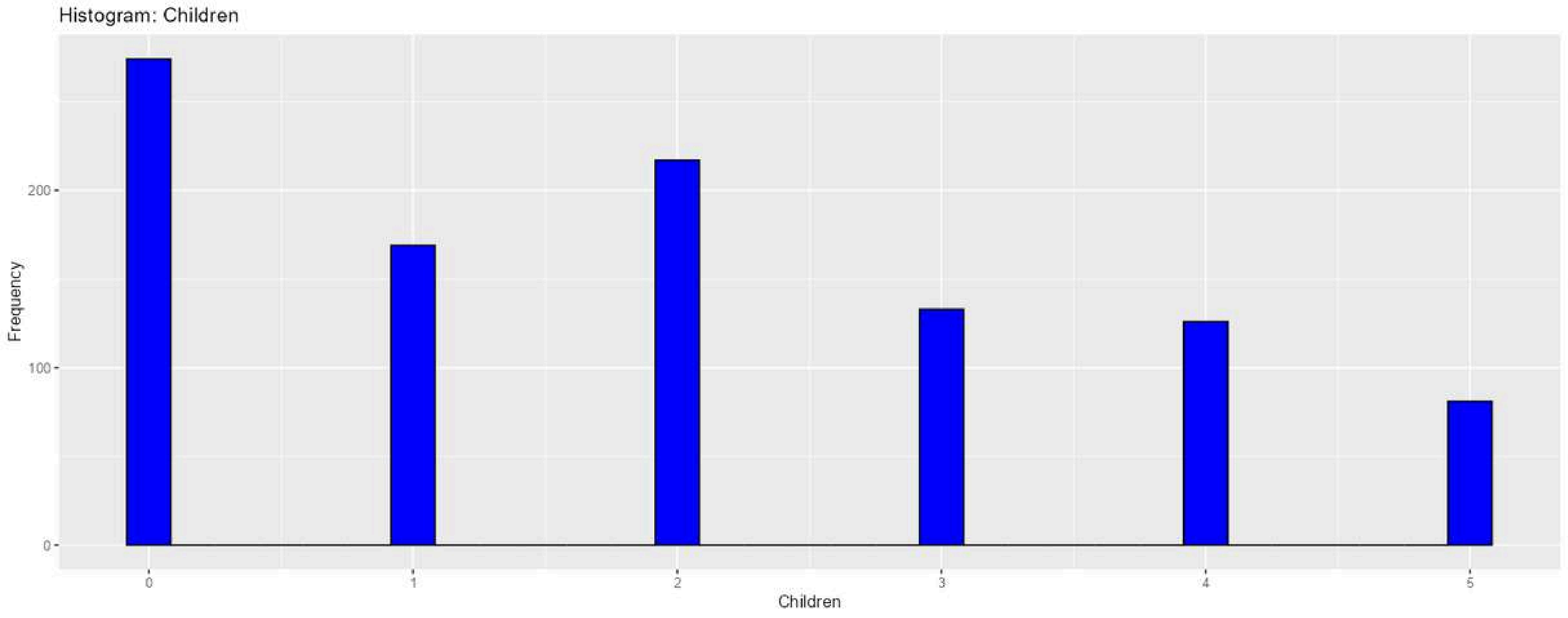
```
 Marital.Status       Gender          Income          Children       Education       Occupation
 Min.   :0.000   Min.   :0.000   Min.   : 10000   Min.   :0.000   Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   1st Qu.:0.000   1st Qu.: 30000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:1.000
 Median :0.000   Median :1.000   Median : 60000   Median :2.000   Median :2.000   Median :3.000
 Mean   :0.458   Mean   :0.511   Mean   : 56290   Mean   :1.911   Mean   :1.631   Mean   :2.259
 3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.: 70000   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:4.000
 Max.   :1.000   Max.   :1.000   Max.   :170000   Max.   :5.000   Max.   :4.000   Max.   :4.000
   Home.Owner         Cars       Commute.Distance     Region           Age        Purchased.Bike
 Min.   :0.000   Min.   :0.000   Min.   :0.000    Min.   :0.000   Min.   :25.00   Min.   :0.000
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.000    1st Qu.:0.000   1st Qu.:35.00   1st Qu.:0.000
 Median :1.000   Median :1.000   Median :1.000    Median :1.000   Median :43.00   Median :0.000
 Mean   :0.686   Mean   :1.451   Mean   :1.645    Mean   :0.892   Mean   :44.17   Mean   :0.481
 3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.:3.000    3rd Qu.:1.000   3rd Qu.:52.00   3rd Qu.:1.000
 Max.   :1.000   Max.   :4.000   Max.   :4.000    Max.   :2.000   Max.   :89.00   Max.   :1.000
```

Visualization

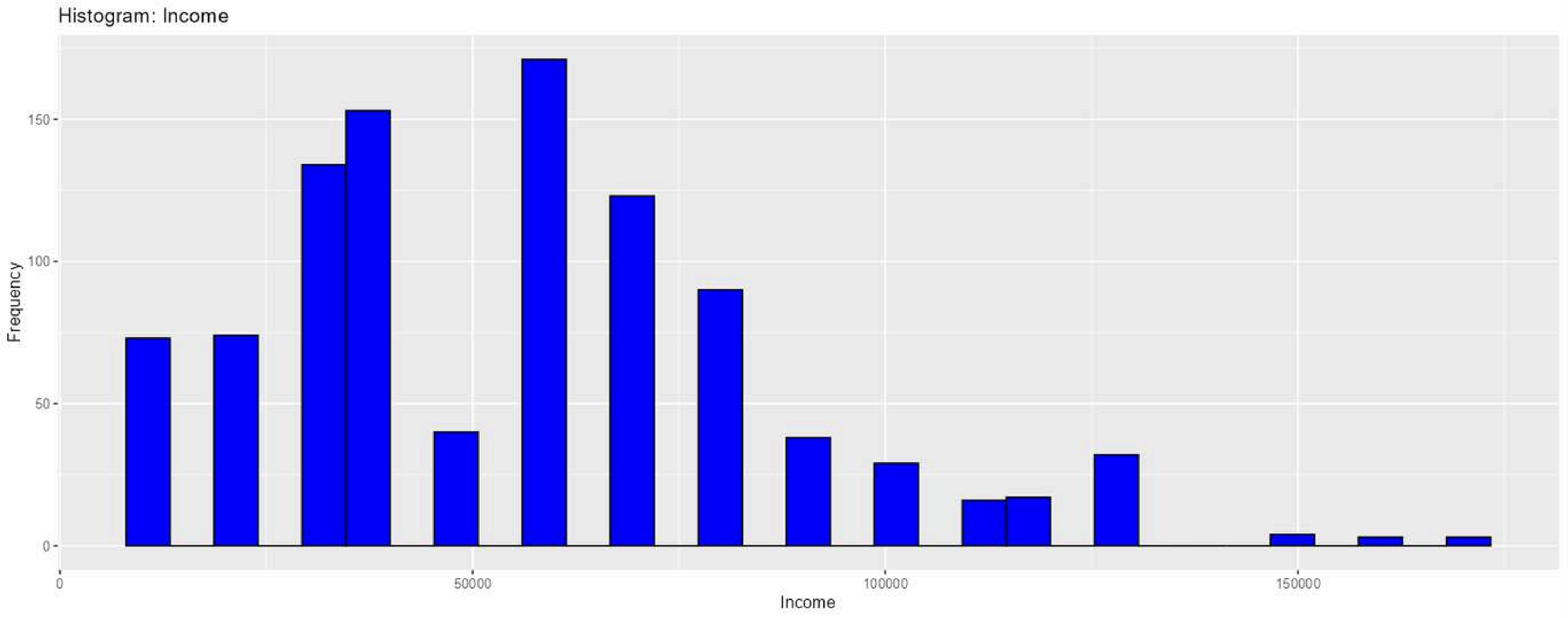
HISTOGRAM



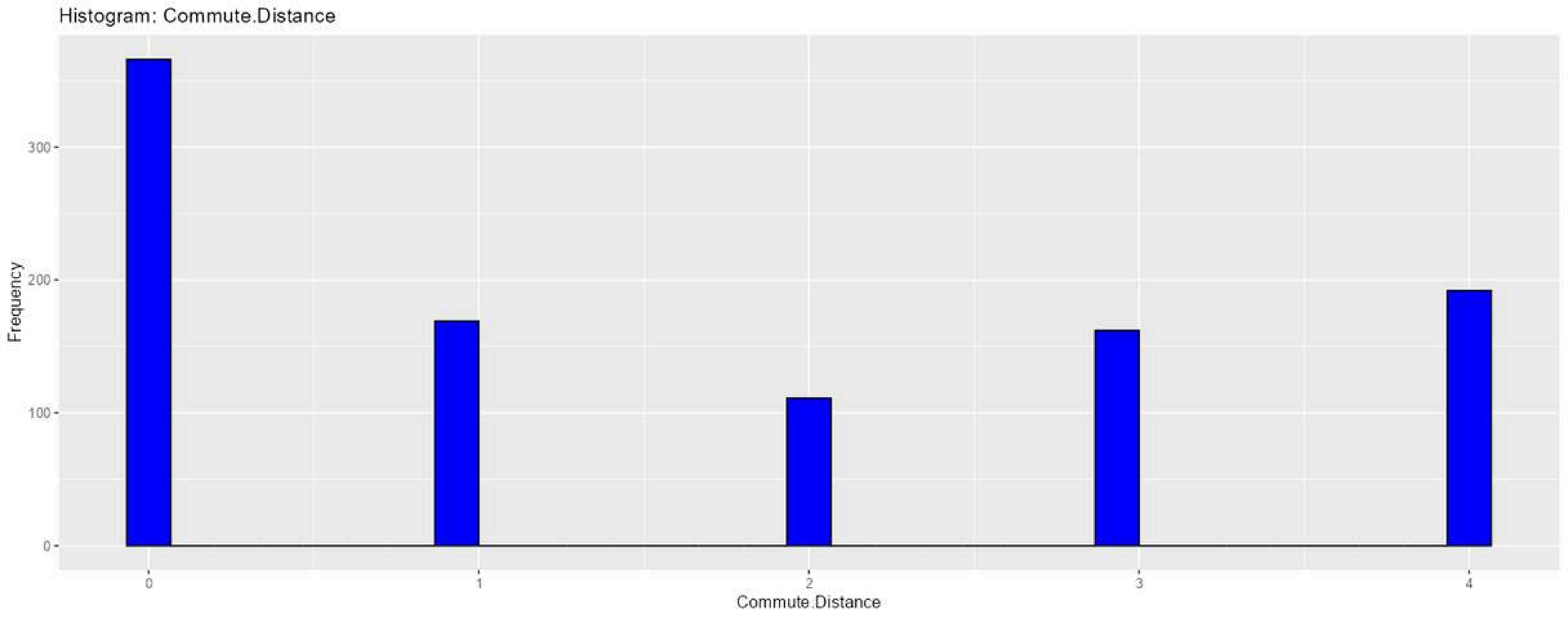




HISTOGRAM

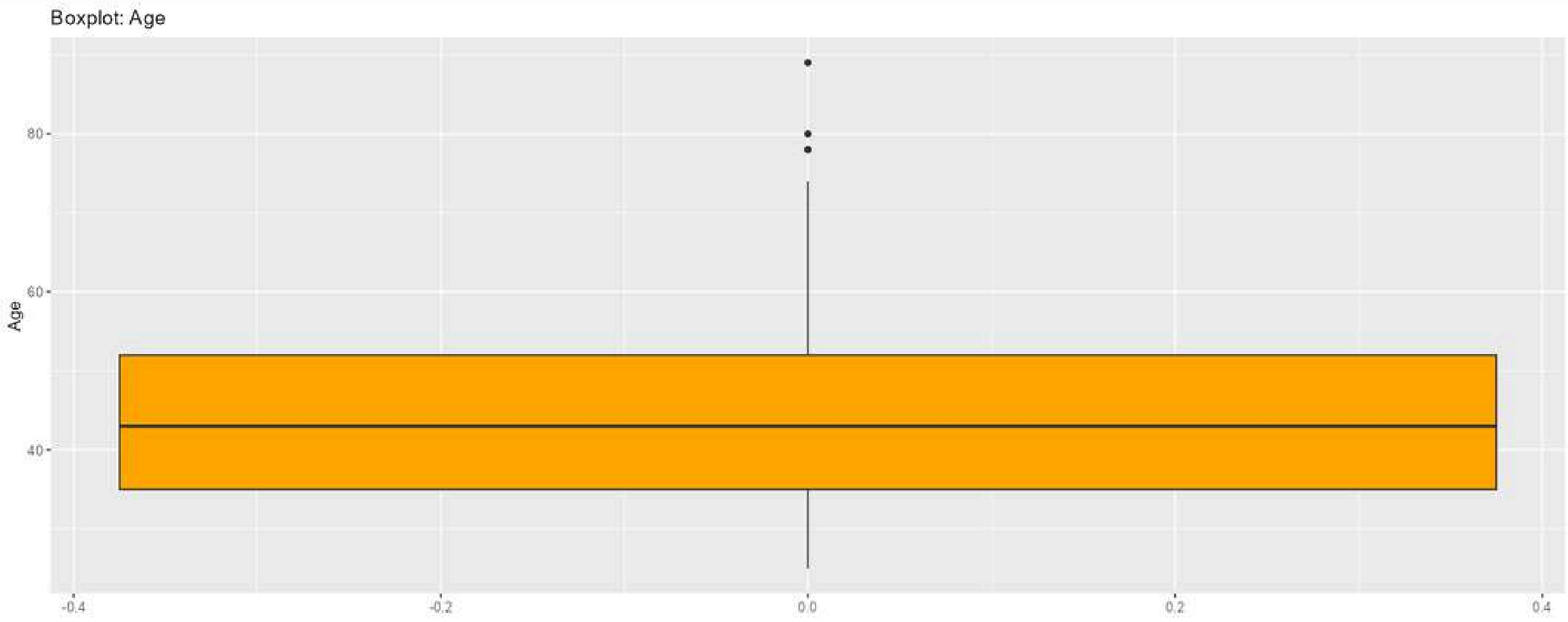


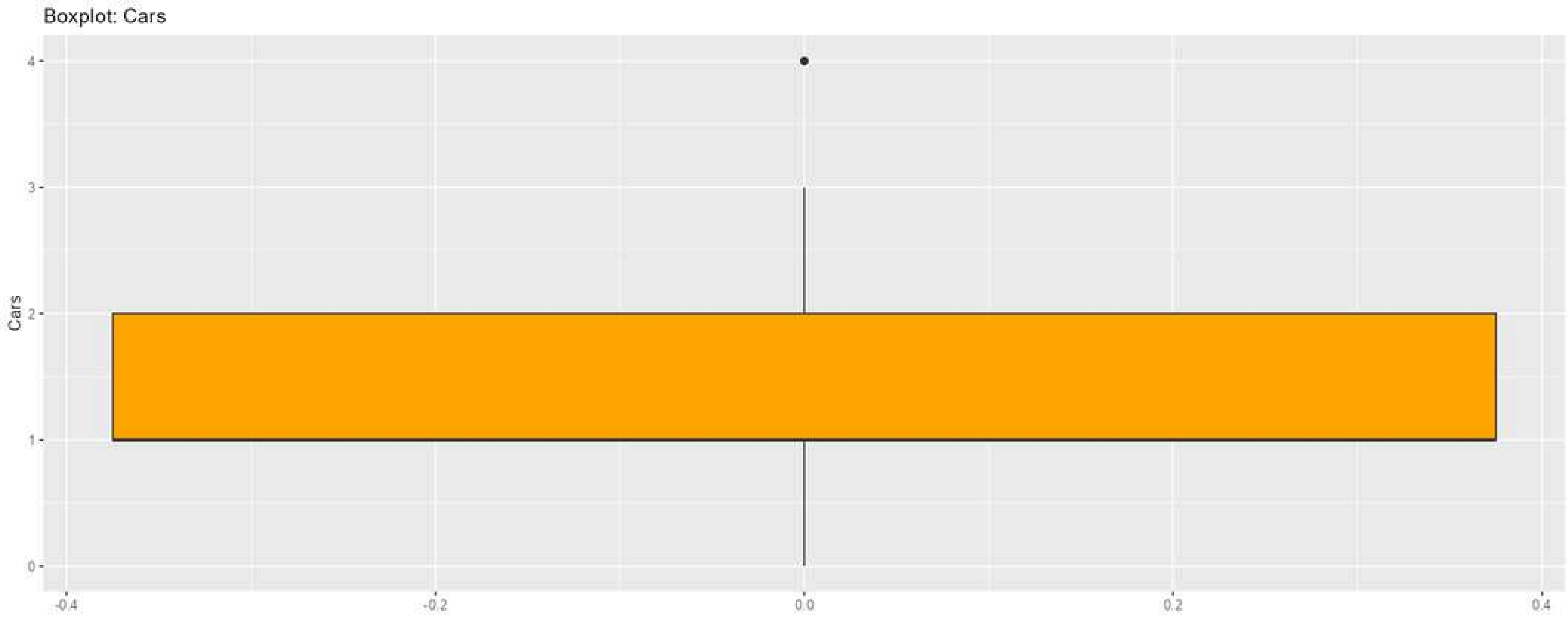
Histogram: Income
Frequency
150
100
50
0
0
50000
100000
150000
Income

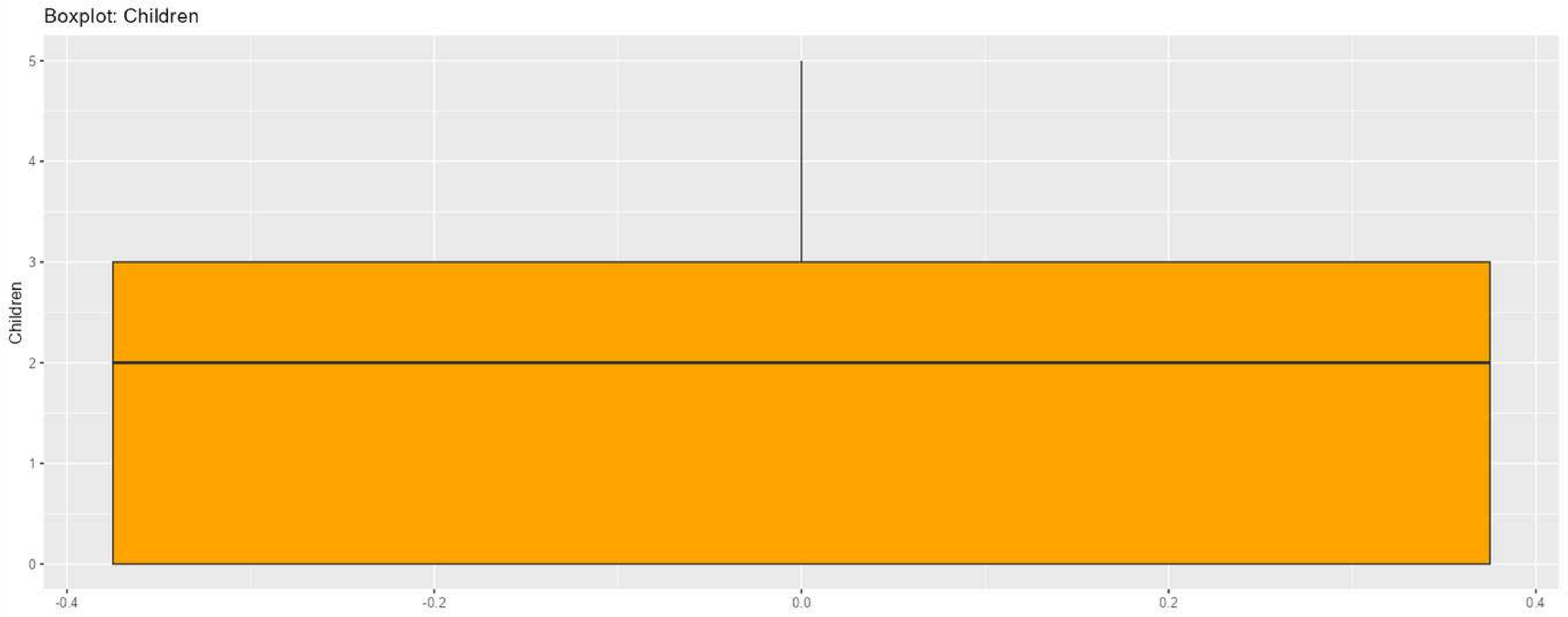


Histogram: Commute.Distance
Frequency
300
200
100
0
0
1
2
3
4
Commute.Distance

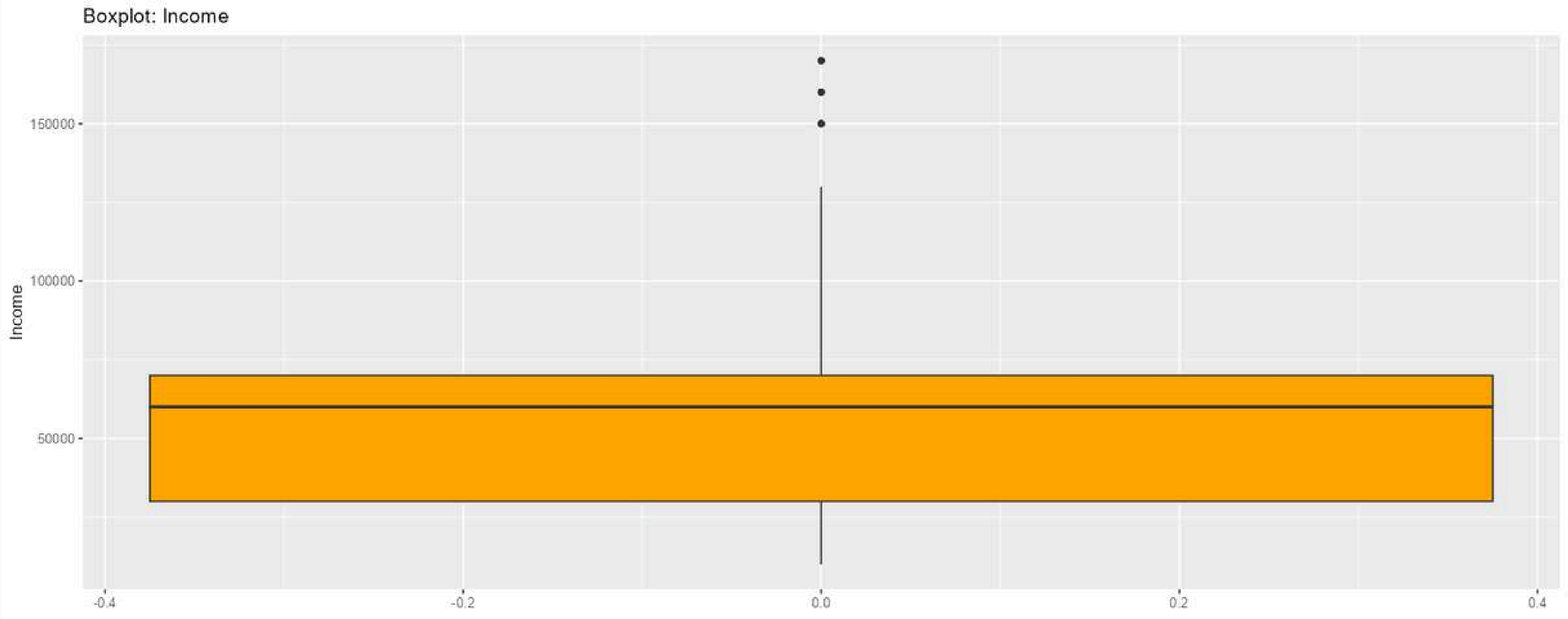
BOX PLOT

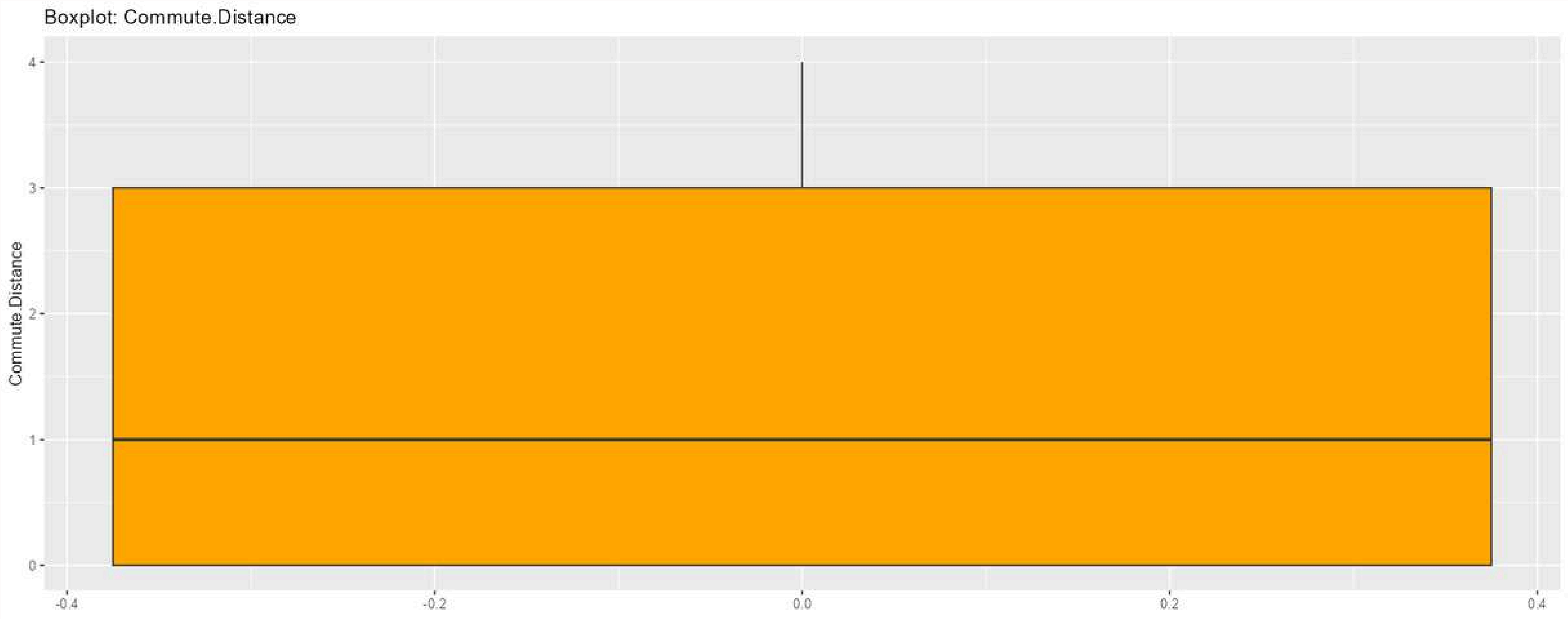

Boxplot: Age
Age
80
60
40
-0.4
-0.2
0.0
0.2
0.4


Boxplot: Cars
Cars
4
3
2
1
0
-0.4
-0.2
0.0
0.2
0.4

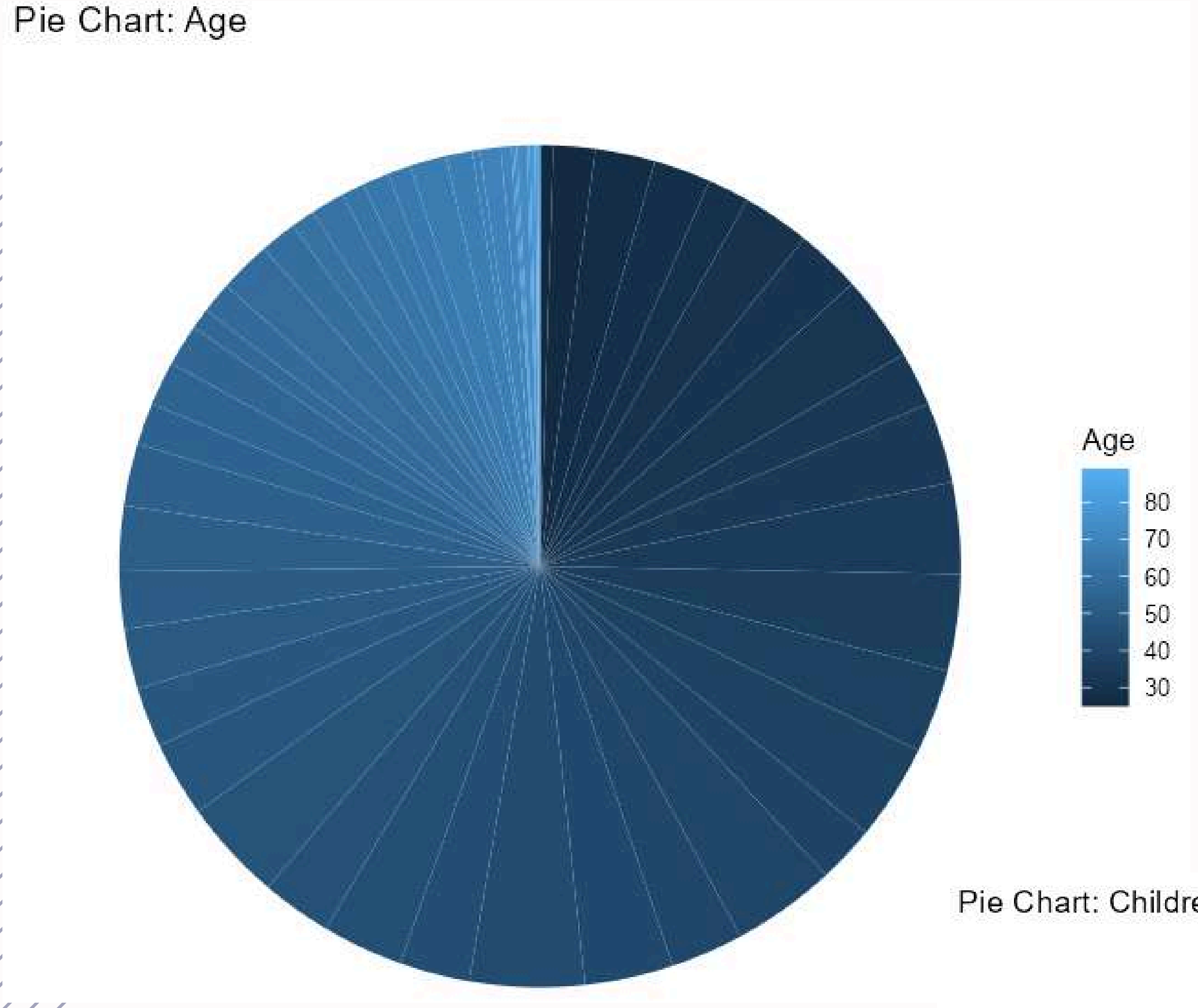

Boxplot: Children
Children
5
4
3
2
1
0
-0.4
-0.2
0.0
0.2
0.4

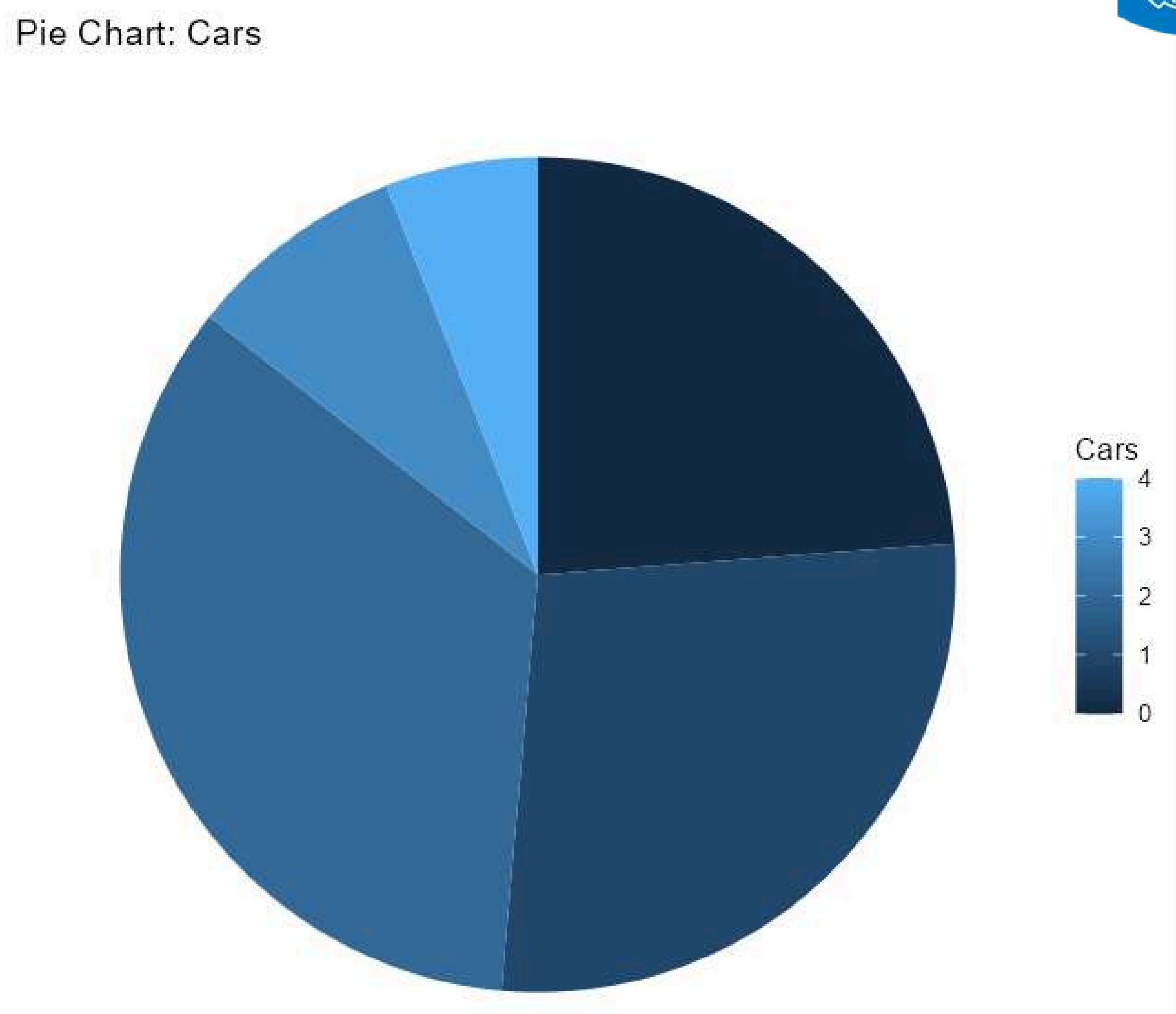
BOX PLOT

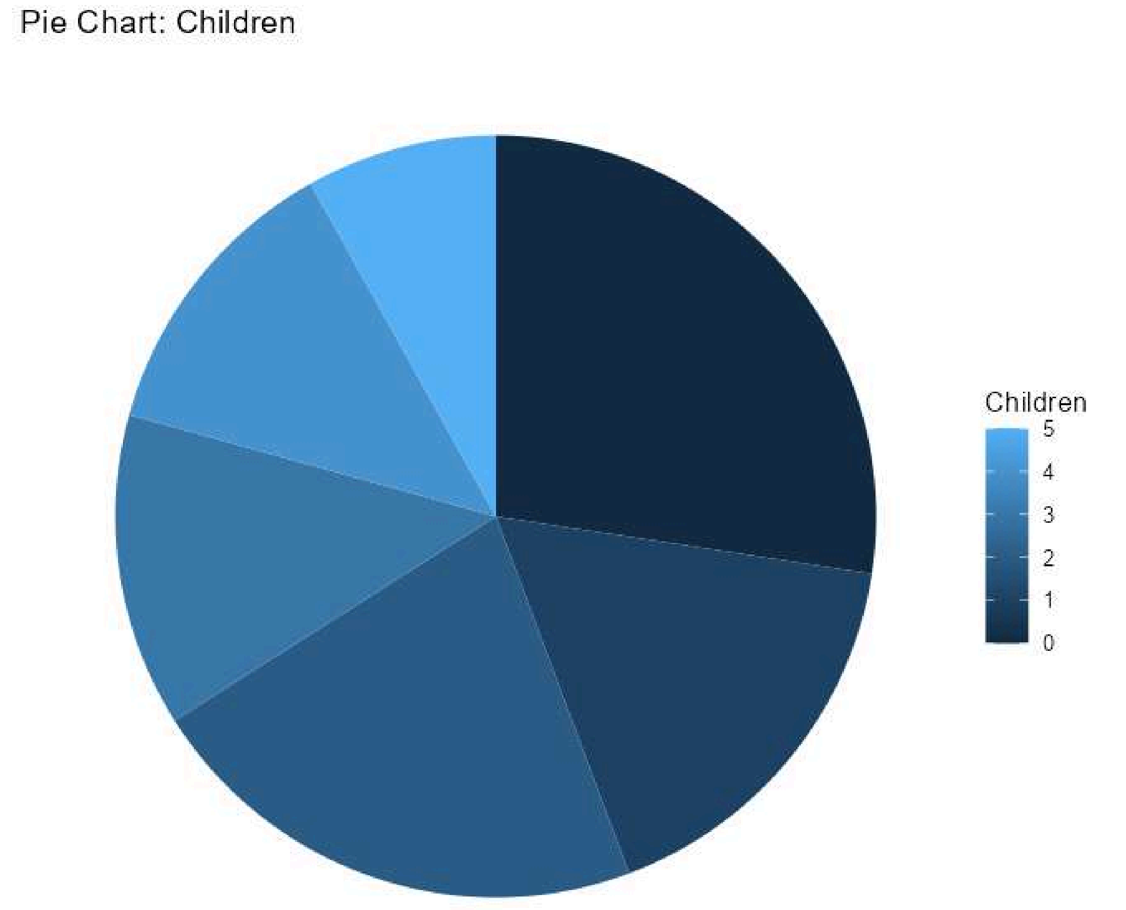

Boxplot: Income
150000
100000
50000
Income
-0.4
-0.2
0.0
0.2
0.4


Boxplot: Commute.Distance
4
3
2
1
0
Commute.Distance
-0.4
-0.2
0.0
0.2
0.4

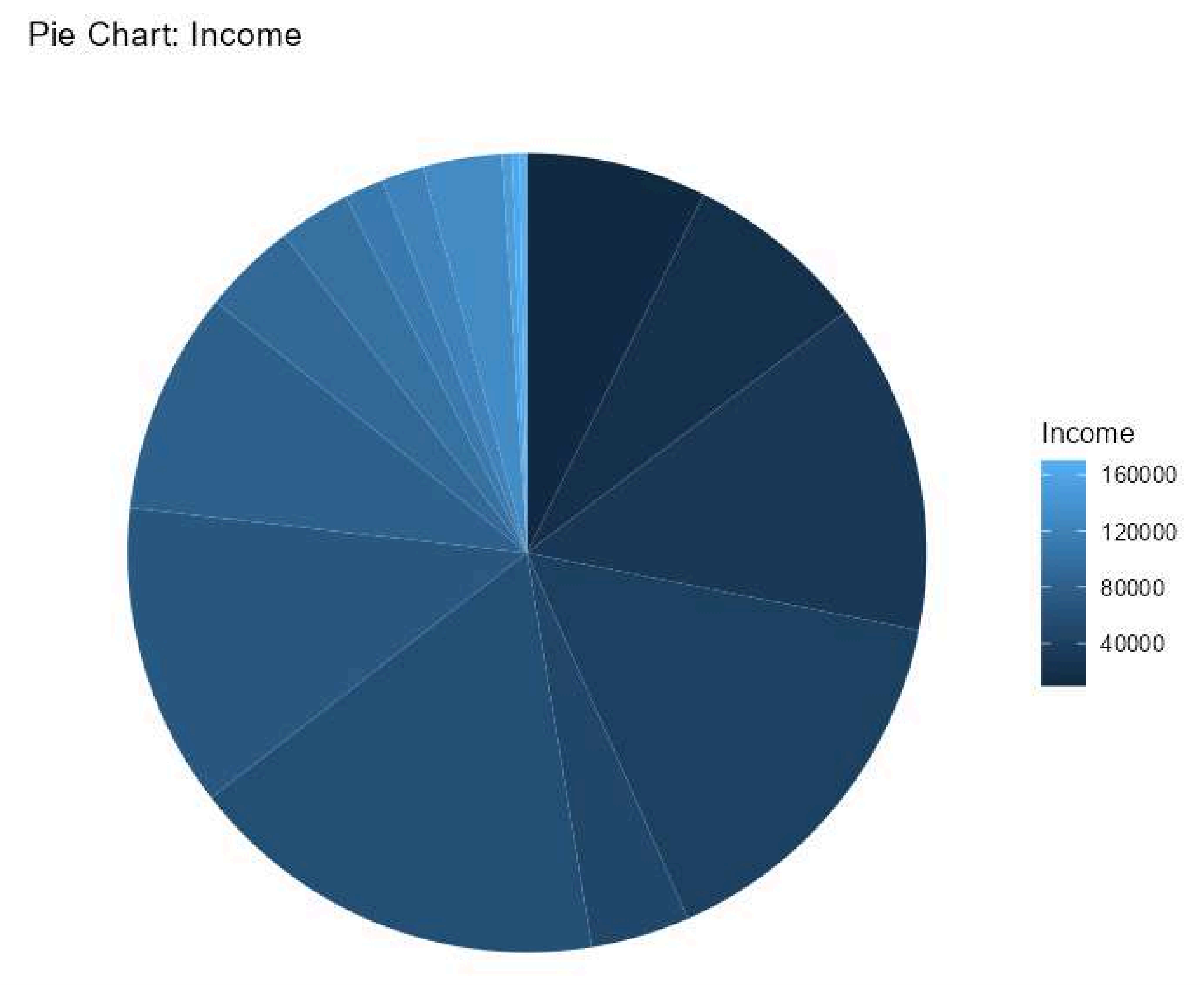
PIE CHART

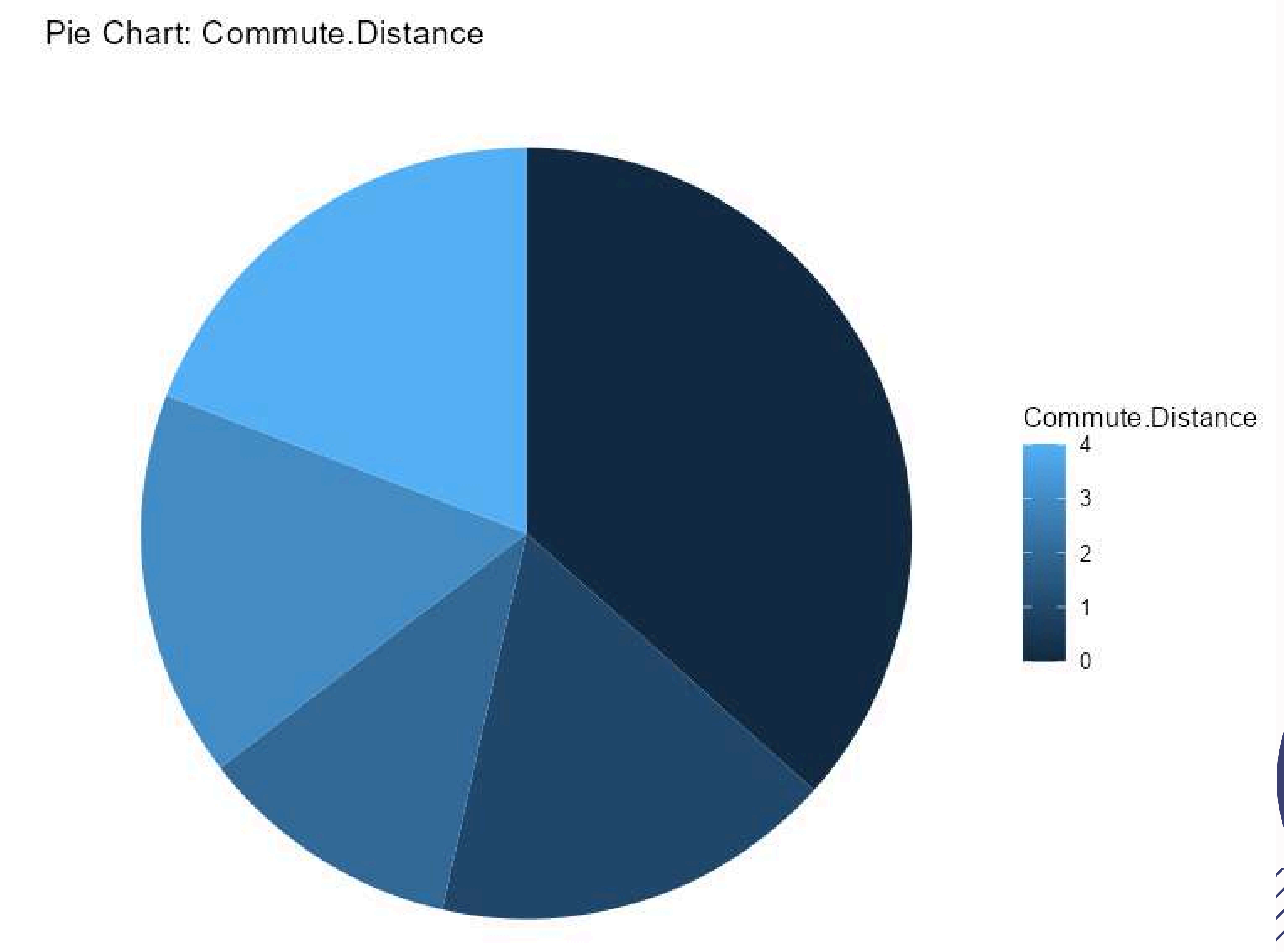

Pie Chart: Age
Age
80
70
60
50
40
30


Pie Chart: Cars
Cars
4
3
2
1
0


Pie Chart: Children
Children
5
4
3
2
1
0

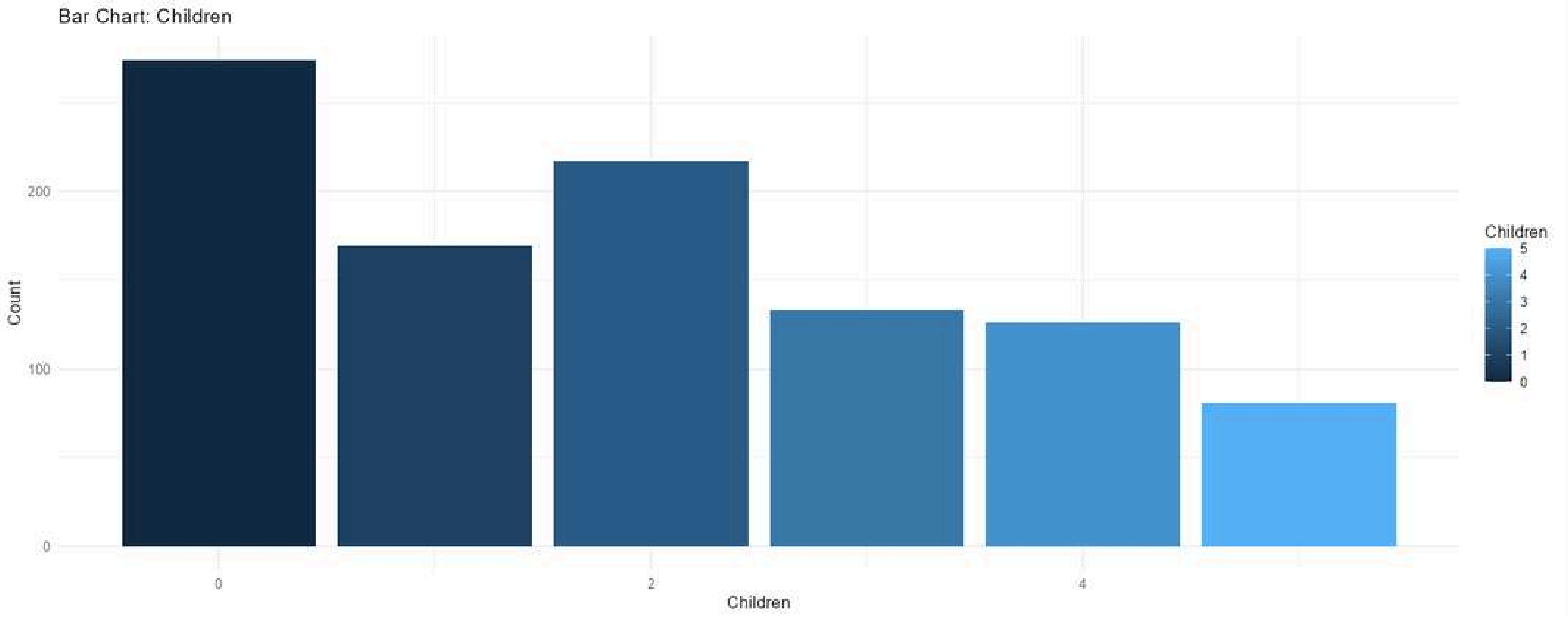
PIE CHART

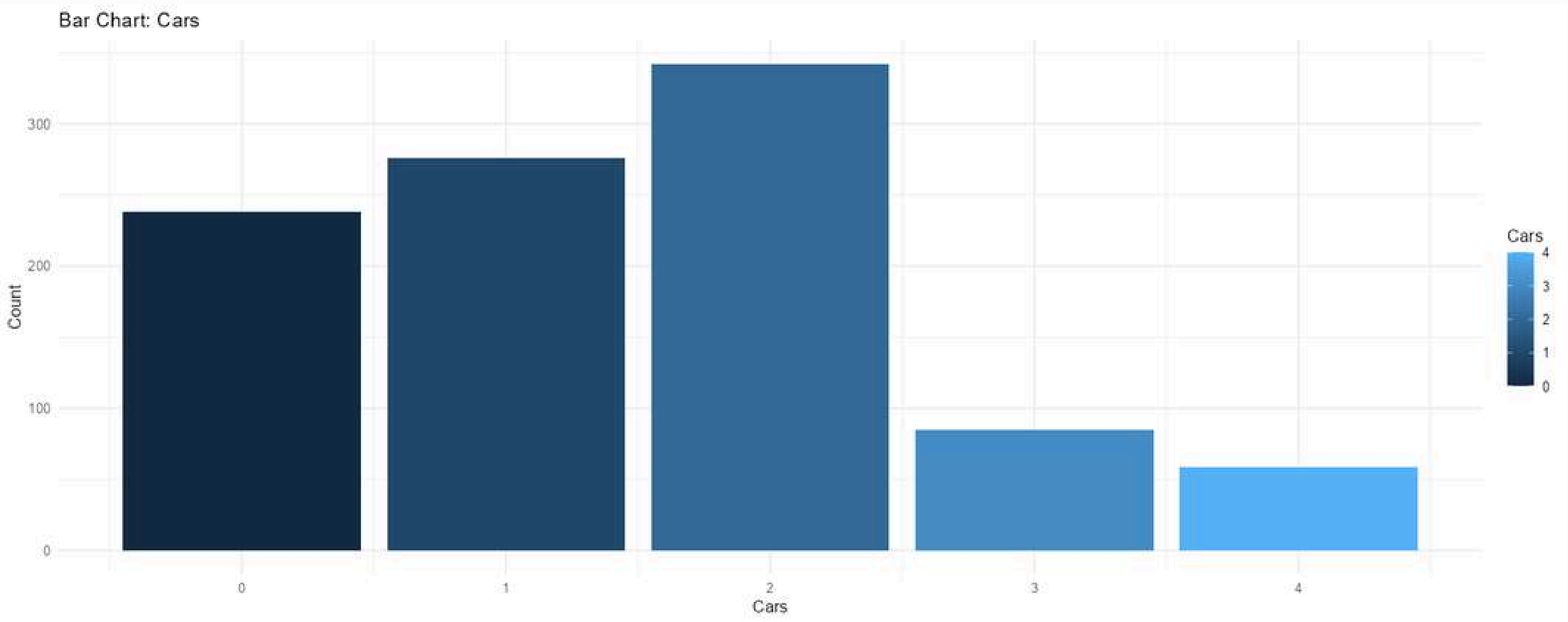

Pie Chart: Income
Income
160000
120000
80000
40000

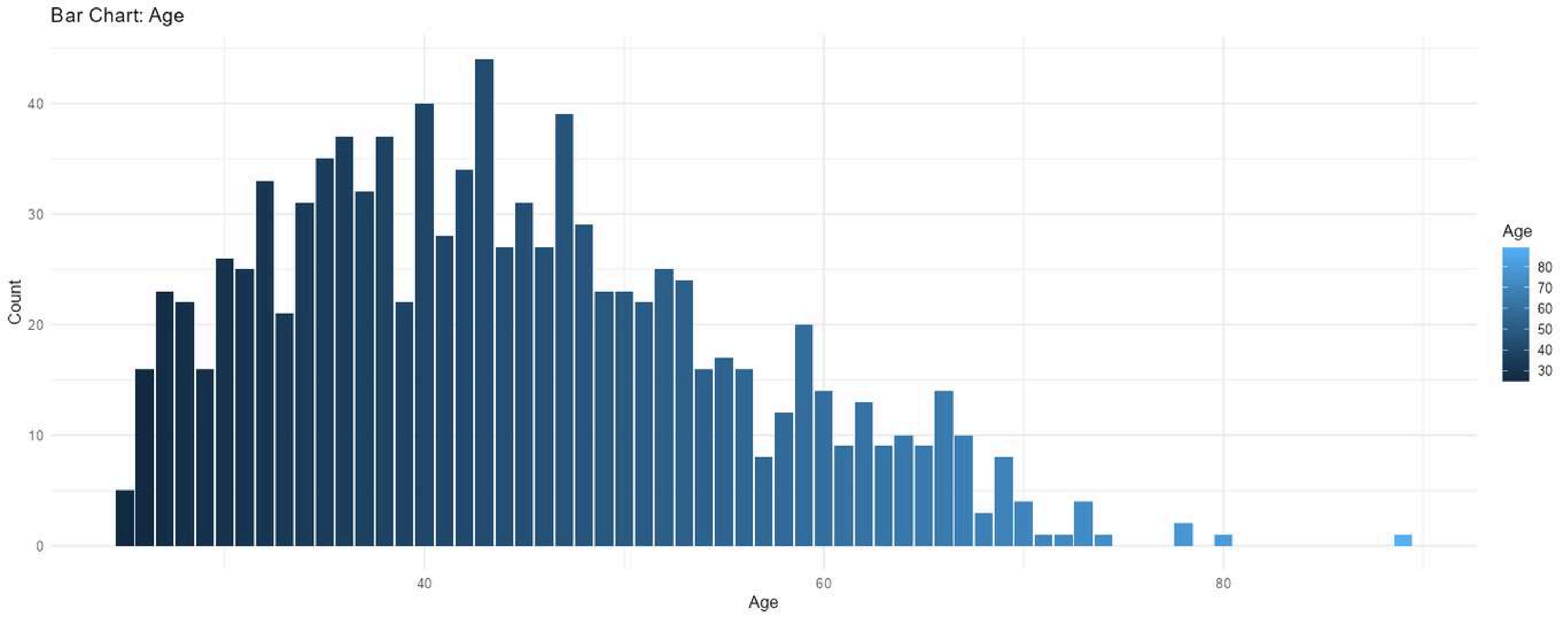

Pie Chart: Commute.Distance
Commute.Distance
4
3
2
1
0

BAR CHART

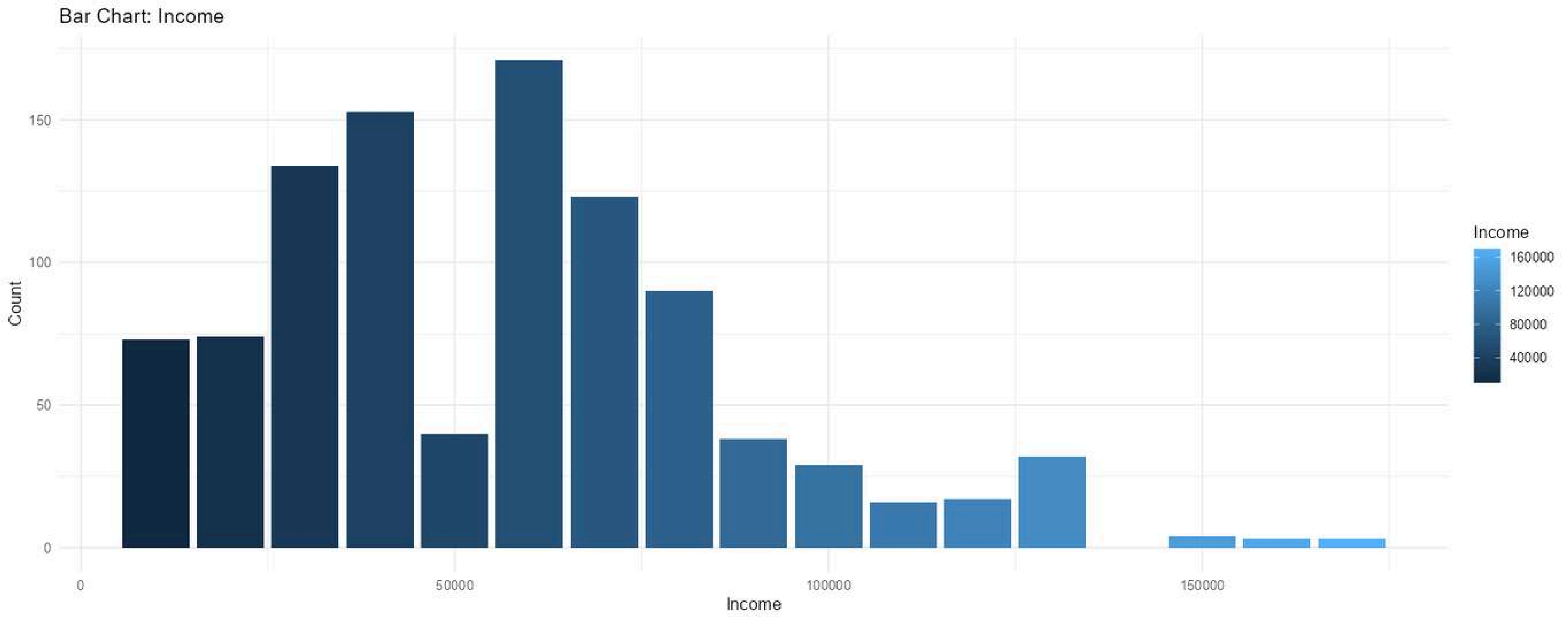
ITS
Institut
Teknologi
Sepuluh Nopember

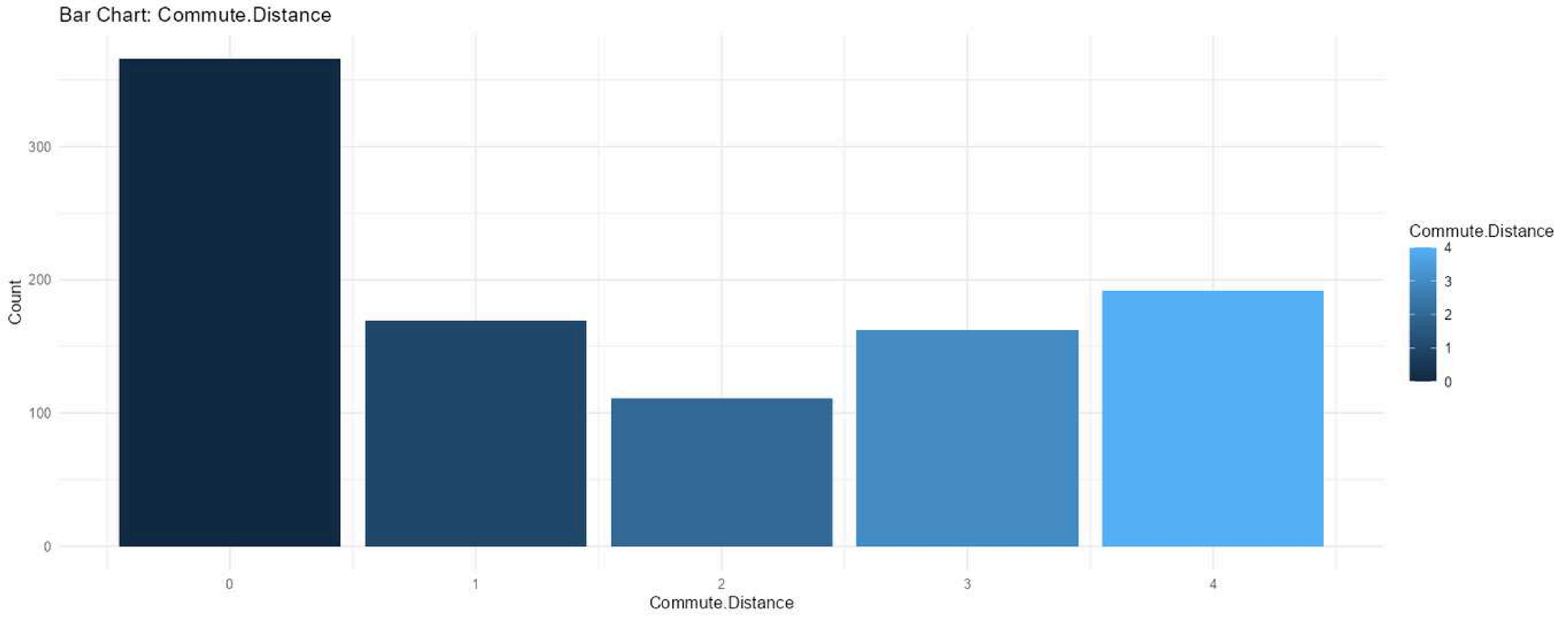




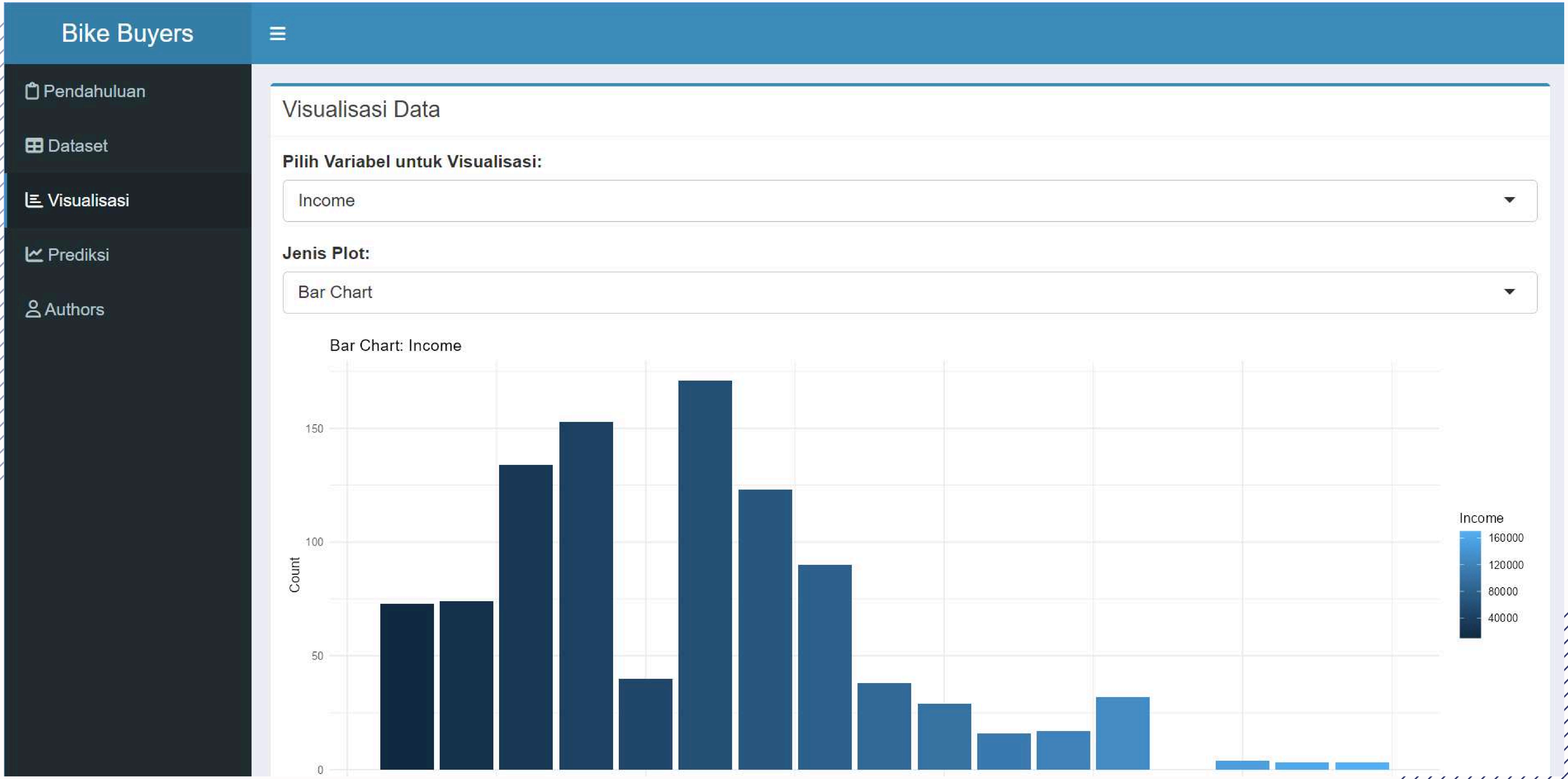


BAR CHART


Bar Chart: Income
Count
150
100
50
0
0
50000
100000
150000
Income
Income
160000
120000
80000
40000


Bar Chart: Commute.Distance
Count
300
200
100
0
0
1
2
3
4
Commute.Distance
Commute.Distance
4
3
2
1
0

Visualisasi Dashboard



Link Demo Rshiny dan Dataset Bike Buyers

DEMO RSHINY DATASET BIKE BUYERS

DATASET BIKE BUYERS

FEATURE SELECTION / EXTRACTION

Cek Data

```
data=bike
data.head()
```

	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	Married	Female	40000.0	1.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	Europe	42.0	0
1	Married	Male	30000.0	3.0	Partial College	Clerical	Yes	1.0	0-1 Miles	Europe	43.0	0
2	Married	Male	80000.0	5.0	Partial College	Professional	No	2.0	2-5 Miles	Europe	60.0	0
3	Single	Male	70000.0	0.0	Bachelors	Professional	Yes	1.0	5-10 Miles	Pacific	41.0	1
4	Single	Male	30000.0	0.0	Bachelors	Clerical	No	0.0	0-1 Miles	Europe	36.0	1

Karena pada data masih terdapat beberapa variabel yang berupa kategorikal tetapi belum berupa numerik, maka dilakukan label encoder untuk memberikan kode (angka) pada variabel kategorikal tersebut.

Label Encoder

```python
# Membuat objek LabelEncoder
label_encoder = LabelEncoder()

# Melakukan label encoding untuk setiap kolom kategorikal
for column in data.select_dtypes(include=['object']).columns:
    data[column] = label_encoder.fit_transform(data[column])
```

```python
data.head()
```

	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	0	0	40000.0	1.0	0	4	1	0.0	0	0	42.0	0
1	0	1	30000.0	3.0	3	0	1	1.0	0	0	43.0	0
2	0	1	80000.0	5.0	3	3	0	2.0	3	0	60.0	0
3	1	1	70000.0	0.0	0	3	1	1.0	4	2	41.0	1
4	1	1	30000.0	0.0	0	0	0	0.0	0	0	36.0	1

Variabel kategorikal sudah berubah sesuai label encoder yang dibentuk.

Label Encoder

```python
# Membuat objek LabelEncoder
label_encoder = LabelEncoder()

# Melakukan label encoding untuk setiap kolom kategorikal
for column in data.select_dtypes(include=['object']).columns:
    data[column] = label_encoder.fit_transform(data[column])
```

```python
data.head()
```

	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	0	0	40000.0	1.0	0	4	1	0.0	0	0	42.0	0
1	0	1	30000.0	3.0	3	0	1	1.0	0	0	43.0	0
2	0	1	80000.0	5.0	3	3	0	2.0	3	0	60.0	0
3	1	1	70000.0	0.0	0	3	1	1.0	4	2	41.0	1
4	1	1	30000.0	0.0	0	0	0	0.0	0	0	36.0	1

Variabel kategorikal sudah berubah sesuai label encoder yang dibentuk.

Cek Info Data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Marital Status    1000 non-null   int64  
 1   Gender            1000 non-null   int64  
 2   Income            1000 non-null   float64
 3   Children          1000 non-null   float64
 4   Education         1000 non-null   int64  
 5   Occupation        1000 non-null   int64  
 6   Home Owner        1000 non-null   int64  
 7   Cars              1000 non-null   float64
 8   Commute Distance  1000 non-null   int64  
 9   Region            1000 non-null   int64  
 10  Age               1000 non-null   float64
 11  Purchased Bike    1000 non-null   int64  
```

Setelah mengatasi missing value dan melakukan juga label encoder, semua variabel data non-null dengan jumlah yang sama yaitu 1000 observasi

Membagi Variabel X dan Y

```
X = data.drop('Purchased Bike', axis=1)
y = data['Purchased Bike']
```

Variabel X nya adalah semua variabel selain Purchased Bike (Marital Status, Gender, Income, CHildren, Education, Occupation, Home Owner, Cars, Commute Distance, Region, dan Age). Sementara Variabel Y nya adalah **Purchased Bike**

Membagi Data (Training-Testing)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Least Absolute Shrinkage and Selection Operator (LASSO) Regression

```python
#alpha sebagai parameter regulasi
lasso = Lasso(alpha=0.01)

# Memasukkan model pada training data
lasso.fit(X_train, y_train)

# Menggunakan SelectFromModel untuk memilih fitur dengan koefisien non-nol
selector = SelectFromModel(lasso, max_features=5)  # Memilih 5 fitur terbaik
X_train_selected = selector.transform(X_train)
X_test_selected = selector.transform(X_test)
```

Didapatkan 5 Fitur Terpilih:

```python
selected_features = X.columns[selector.get_support()]
print(f"Fitur terpilih: {selected_features}")
```

```
Fitur terpilih: Index(['Marital Status', 'Children', 'Education', 'Cars', 'Region'], dtype='object')
```

Least Absolute Shrinkage and Selection Operator (LASSO) Regression

Menampilkan koefisien LASSO regression:

```
coefficients = pd.Series(lasso.coef_, index=X.columns)
print("Koefisien Lasso untuk setiap fitur:")
print(coefficients)
```

```
Koefisien Lasso untuk setiap fitur:
Marital Status      0.071755
Gender              0.000000
Income              0.000003
Children           -0.011385
Education          -0.015887
Occupation         -0.003756
Home Owner          0.000000
Cars               -0.120735
Commute Distance   -0.008710
Region              0.039944
Age                -0.001949
dtype: float64
```

RFE (Recursive Feature Elimination)

```python
# Buat objek RandomForestClassifier
clf_rf_3 = RandomForestClassifier()

# Membuat objek RFE dan melakukan seleksi fitur
rfe = RFE(estimator=clf_rf_3, n_features_to_select=5, step=1)
rfe = rfe.fit(X, y)
```

```python
print('Five Best Features :', X.columns[rfe.support_])
```

```
Five Best Features : Index(['Income', 'Children', 'Cars', 'Commute Distance', 'Age'], dtype='object')
```

```python
# Menampilkan feature importance
for i, feature in enumerate(X.columns):
    print(f"{feature}: {feature_importance[i]:.4f}")
```

```
Marital Status: 0.0498
Gender: 0.0490
Income: 0.1260
Children: 0.1052
Education: 0.0723
Occupation: 0.0557
Home Owner: 0.0335
Cars: 0.0968
Commute Distance: 0.1072
Region: 0.0528
Age: 0.2517
```

Didapatkan 5 Fitur Teratas:

```python
# Memilih 5 fitur teratas berdasarkan feature importance
XRFE_features = X.columns[np.argsort(feature_importance)[::-1][:5]]
print(f"5 Fitur Terpenting: {XRFE_features}")
```

```
5 Fitur Terpenting: Index(['Age', 'Income', 'Commute Distance', 'Children', 'Cars']
```

CLASSIFICATION

Regresi Logistik

```
#klasifikasi menggunakan regresi logistik

from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.impute import SimpleImputer

# Use X instead of X.columns
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Logistic Regression model
logreg_model = LogisticRegression()
logreg_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = logreg_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.54
              precision    recall  f1-score   support

           0       0.56      0.65      0.60       106
           1       0.51      0.41      0.46        94

    accuracy                           0.54       200
   macro avg       0.53      0.53      0.53       200
weighted avg       0.54      0.54      0.53       200
```

Didapatkan nilai akurasi sebesar 0.54

Naive Bayes

```
#klasifikasi menggunakan naive bayes

from sklearn.naive_bayes import GaussianNB

#klasifikasi menggunakan Naive Bayes
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)
accuracy_nb = accuracy_score(y_test, y_pred_nb)

print(f"Accuracy Naive Bayes: {accuracy_nb}")
print(classification_report(y_test,y_pred_nb))
print(confusion_matrix(y_test,y_pred_nb))
```

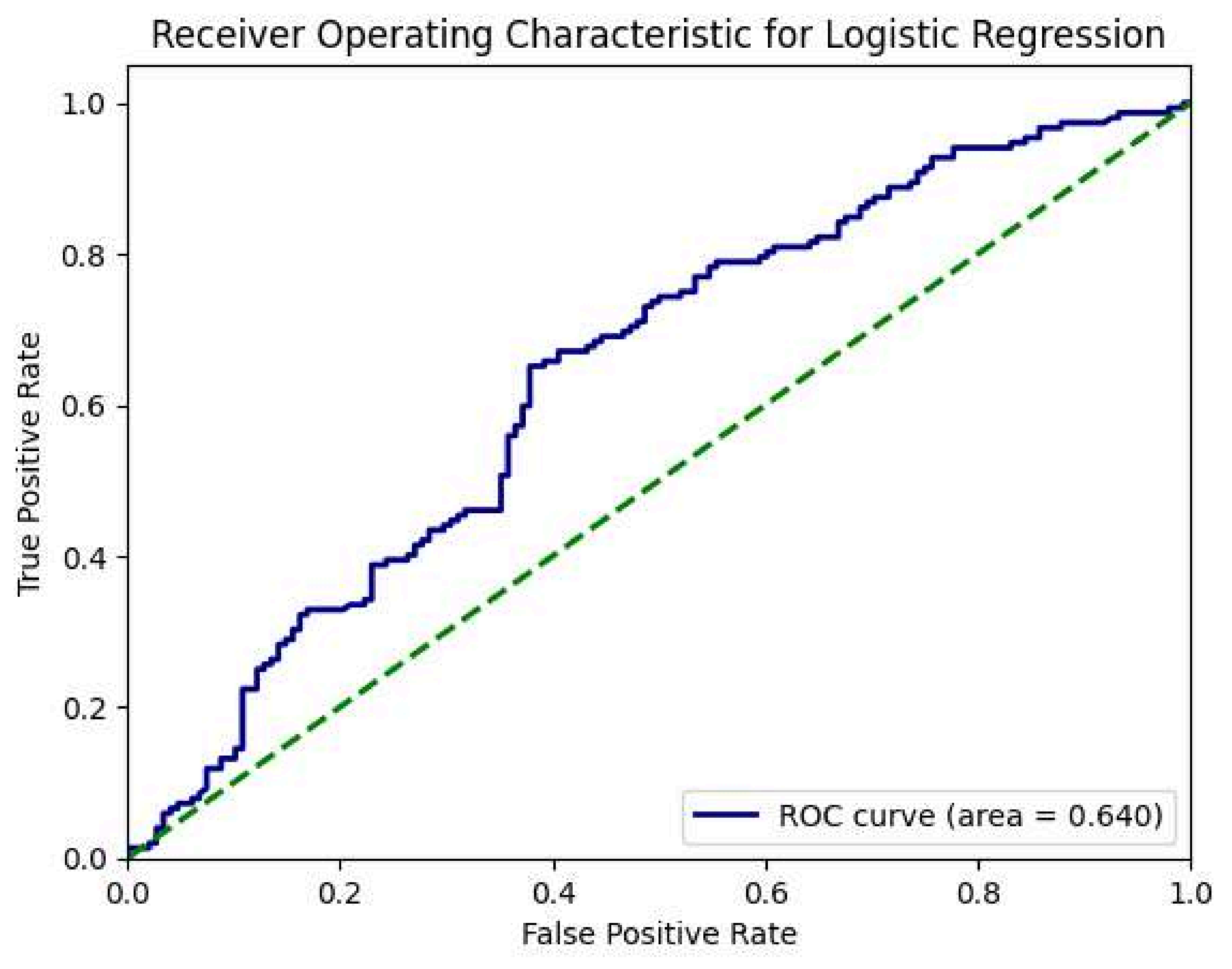
```
Accuracy Naive Bayes: 0.555
              precision    recall  f1-score   support

           0       0.58      0.60      0.59       106
           1       0.53      0.50      0.51        94

    accuracy                           0.56       200
   macro avg       0.55      0.55      0.55       200
weighted avg       0.55      0.56      0.55       200
```

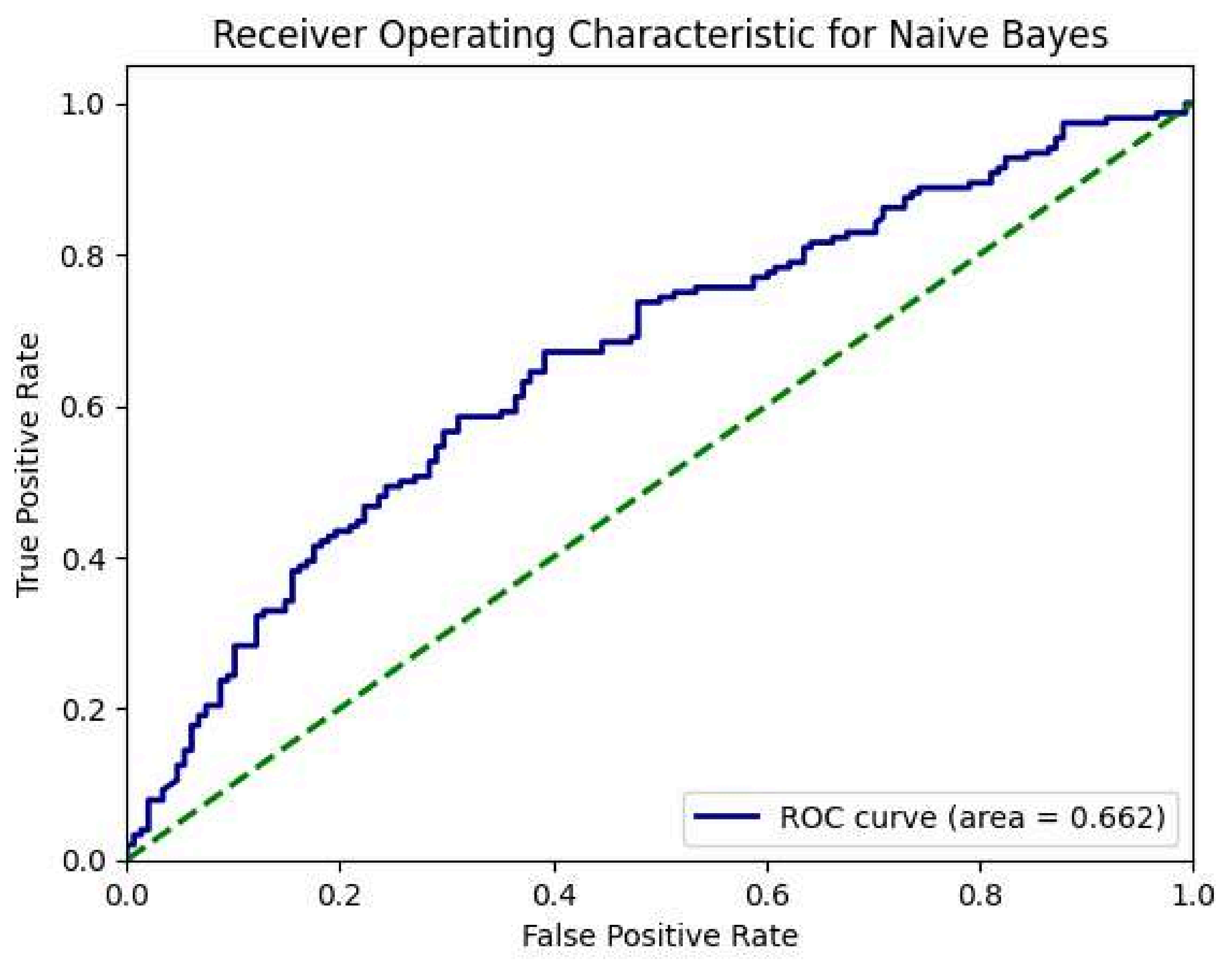
Didapatkan nilai akurasi sebesar 0.555

Jika dibandingkan dengan nilai akurasi dari klasifikasi menggunakan Regresi Logistik hasilnya tidak jauh beda.

TRAINING-TESTING

Repeated Holdout

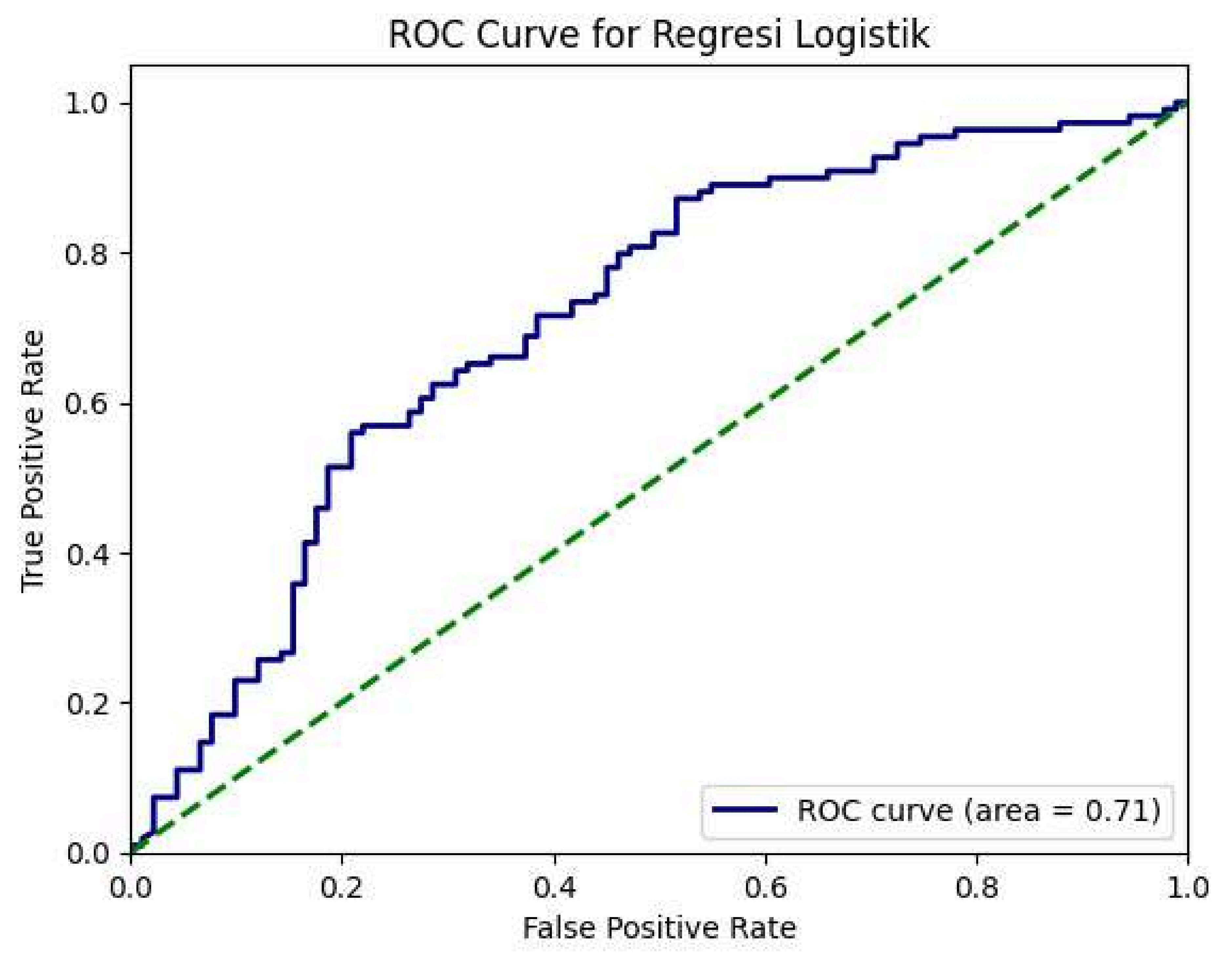





```
Akurasi: 0.5666666666666667
ROC AUC: 0.6396915007112376
Sensitivitas (Recall): 0.4473684210526316
Spesifisitas: 0.6891891891891891
```

```
Akurasi: 0.63
ROC AUC: 0.6623177453769559
Sensitivitas (Recall): 0.5657894736842105
Spesifisitas: 0.6959459459459459
```

K-Fold (K=5) Regresi Logistik

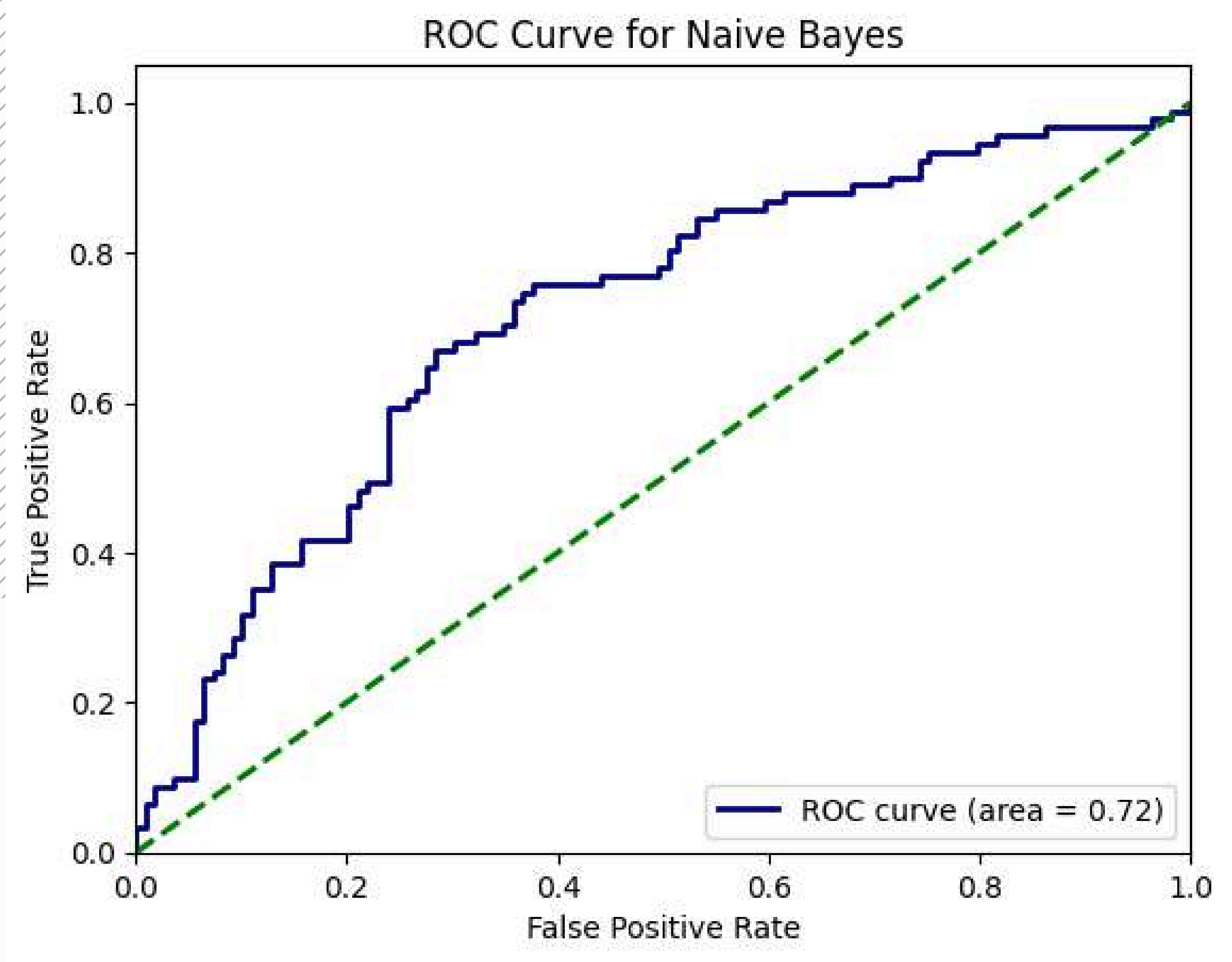


Akurasi: 0.66

```
Rata-rata ROC AUC: 0.6567113409341445
Rata-rata Sensitivitas (Recall): 0.5754624247182318
Rata-rata Spesifisitas: 0.681683285556403
Rata-rata Matriks Konfusi:
[[70.6 33.2]
 [40.8 55.4]]
```

K-Fold (K=5)

Naive Bayes



Akurasi: 0.69

```
Rata-rata ROC AUC: 0.6574954561338485
Rata-rata Sensitivitas (Recall): 0.6160481147293074
Rata-rata Spesifisitas: 0.6333164671575611
Rata-rata Matriks Konfusi:
[[65.4 38.4]
 [37.  59.2]]
```

PERBANDINGAN DAN PEMILIHAN METODE TERBAIK

Jenis Training-Testing	Metode	Akurasi	Sensitifitas	Spesifitas	ROC AUC
Repeated Holdout	Regresi Logistik	0.566	0.447	0.689	0.639
	Naive Bayes	0.63	0.566	**0.696**	**0.662**
K-Fold (k=5)	Regresi Logistik	0.66	0.575	0.682	0.657
	Naive Bayes	**0.69**	**0.616**	0.633	0.657

K-Fold (K=5) dengan metode Naive Bayes dipilih sebagai metode terbaik karena memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan metode yang lain.

KESIMPULAN

Berdasarkan analisis yang telah dilakukan dengan menggunakan 2 Metode Klasifikasi, yaitu Regresi Logistik dan Naive Bayes dimana pada masing-masing menggunakan 2 tipe training-testing didapatkan metode terbaik, yaitu **naive bayes** dengan K-Fold (K=5).

Dengan tingkat akurasi sebesar 0.69, dimana model dapat mengklasifikasikan data dengan benar sebesar 69% dan presentasi ini menunjukkan model sudah baik. Di dukung dengan tingkat sensivitas 61.6%, 63.3% tingkat spesifisitas, dan nilai ROC AUC 0.657 dimana artinya model sudah baik dalam membedakan kelas positif dan negatif.

Sehingga secara keseluruhan, nilai parameter di atas sudah mampu menunjukkan bahwa metode Naive Bayes dapat dikatakan baik dalam mengklasifikan data “bike buyers”.

Data Mining dan Visualisasi C

THANK YOU

“Every time you miss your childhood,
RIDE ON A BICYCLE”