Analytical PART:

To meet the requirements, I wrote the analysis from Step 1–6 entirely without any data mining methods, focusing only on data loading, cleaning, transformation, and Exploratory Data Analysis (EDA). After completing the exploratory phase, I applied two new data mining methods: **Random Forest Regression** and **Neural Network Regression (MLPRegressor)**.

Random Forest provided a powerful ensemble-based approach to understand feature importance and predict rental demand, while the Neural Network model captured complex non-linear relationships within the dataset. Together, these models extended the insights gained during EDA into robust predictive analytics, completing a comprehensive analytical workflow using the Bike Sharing dataset.

Why These Two Data Mining Methods Were Selected and How They Support the EDA Objectives:
The Exploratory Data Analysis (EDA) revealed meaningful trends and variable relationships within the Bike Sharing (day.csv) dataset. These insights guided the selection of **Random Forest Regression** and **Neural Network Regression**, as both methods directly extend the patterns discovered during EDA into strong predictive models.

Random Forest Regression- How EDA Led to This Choice
EDA revealed several patterns:

- Rental counts increase with temperature
- Humidity and windspeed strongly affect demand
- Season and weather significantly impact user behavior
- Several variables interact in non-linear ways that may be difficult to capture using simple linear models

These observations suggested the need for a model that:

- Handles **non-linear relationships**
- Automatically captures **feature interactions**
- Provides **feature importance** to validate EDA insights

How Random Forest Supports EDA Objectives
Random Forest helps to:

- Rank which variables influence rentals the most
- Confirm EDA findings about temperature, season, and weather effects
- Produce accurate predictions using many interacting variables
- Handle outliers and non-linear patterns that EDA visualizations hinted at

It bridges the gap between visual trends (EDA) and measurable relationships (predictive modeling).

Neural Network Regression- How EDA Led to This Choice
EDA showed:

- Increasing rentals with temperature
- Sudden drops during poor weather
- Seasonal peaks and troughs
- Several variables combining in complex ways to influence demand

These observations indicated that the dataset may contain **non-linear and layered relationships** that EDA can show visually but cannot model mathematically.
A Neural Network was chosen because it:

- Learns patterns automatically from data
- Models complex relationships beyond simple linear trends
- Detects subtle interactions between temperature, weather, working days, and seasonal changes

How the Neural Network Supports EDA Objectives
The Neural Network model:

- Captures hidden patterns not obvious from charts
- Validates and extends EDA observations through predictive learning
- Learns deeper non-linear relationships suggested by EDA
- Provides an alternative modeling perspective to compare with Random Forest

Python Code:

```
# STEP 1: IMPORT LIBRARIES
import pandas as pd
```

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# ML imports
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler

#STEP 2: LOAD day.csv FILE
uploaded = files.upload()
df = pd.read_csv("day.csv")
print("\nDataset Info:")
print(df.info())
print("\nSummary Statistics:")
print(df.describe())
print("\nMissing Values:")
print(df.isnull().sum())
print("\nDataset Shape:", df.shape)

# STEP 3: DATA CLEANING
# 1. Remove duplicate rows
duplicate_count = df.duplicated().sum()
print("\nDuplicate Rows:", duplicate_count)
if duplicate_count > 0:
    df = df.drop_duplicates()
# 2. Handle missing values using forward-fill
df = df.fillna(method='ffill')
print("\nMissing Values After Cleaning:")
print(df.isnull().sum())
# 3. Convert date column from string to datetime
df['dteday'] = pd.to_datetime(df['dteday'])
# 4.   Drop irrelevant column "instant" if present
if 'instant' in df.columns:
    df.drop(columns=['instant'], inplace=True)
# 5. Rename columns for readability
df.rename(columns={'mnth': 'month', 'yr': 'year'}, inplace=True)
print("\nData Cleaning Completed.")

# STEP 4: DATA EXPLORATION
# Look at skewness and kurtosis of numeric features
print("\nSkewness:")
print(df.skew(numeric_only=True))
print("\nKurtosis:")
print(df.kurt(numeric_only=True))

#  Boxplot to visually inspect outliers
plt.figure(figsize=(10,5))
sns.boxplot(data=df[['temp', 'hum', 'windspeed', 'cnt']])
plt.title("Boxplot of Key Numerical Columns")
plt.show()

# STEP 5: TRANSFORMATION
# Convert normalized temperature values into °C for human readability
df['temp_celsius'] = df['temp'] * 41
df['atemp_celsius'] = df['atemp'] * 50

#  STEP 6: BASIC VISUALIZATIONS (EDA ONLY)
# 1. Distribution of rentals
plt.figure(figsize=(8,5))
```

```python
sns.histplot(df['cnt'], kde=True, bins=30)
plt.title("Distribution of Bike Rentals")
plt.xlabel("Rental Count")
plt.ylabel("Frequency")
plt.show()

# 2. Correlation heatmap
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap="coolwarm", center=0)
plt.title("Correlation heatmap (numeric features)")
plt.show()

# 3. Rentals by season
plt.figure(figsize=(8,5))
sns.boxplot(x='season', y='cnt', data=df, palette='Set2')
plt.title("Bike Rentals Across Seasons")
plt.show()

# 4. Trend of rentals over time
plt.figure(figsize=(12,5))
sns.lineplot(x='dteday', y='cnt', data=df)
plt.title("Bike Rentals Over Time")
plt.show()

# Data Mining method : RANDOM FOREST REGRESSION MODEL

# Select numeric features only for modeling
numeric_df = df.select_dtypes(include=[np.number]).copy()

# X = input features, y = target
X = numeric_df.drop(columns=['cnt'])
y = numeric_df['cnt'].astype(float)

# Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=123)

# Build Random Forest Model
rf = RandomForestRegressor(n_estimators=200, random_state=123, n_jobs=-1)
rf.fit(X_train, y_train)

# Predict on test data
rf_pred = rf.predict(X_test)

# Evaluate performance
rf_mse = mean_squared_error(y_test, rf_pred)
rf_rmse = np.sqrt(rf_mse)
rf_mae = mean_absolute_error(y_test, rf_pred)
rf_r2 = r2_score(y_test, rf_pred)
print("\nRandom Forest Regression Metrics:")
print(f" MAE : {rf_mae:.3f}")
print(f" RMSE: {rf_rmse:.3f}")
print(f" R2  : {rf_r2:.4f}")

# Feature Importance plot
importances = rf.feature_importances_
feat_names = X.columns
idx = np.argsort(importances)
plt.figure(figsize=(8,6))
plt.barh(feat_names[idx], importances[idx])
plt.grid(axis='x', linestyle='--', alpha=0.6)
```

```python
plt.title("Random Forest Feature Importances")
plt.xlabel("Importance")
plt.show()
# Actual vs Predicted scatter plot
plt.figure(figsize=(8,6))
plt.scatter(y_test, rf_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual cnt")
plt.ylabel("Predicted cnt")
plt.title("Random Forest: Actual vs Predicted")
plt.show()

# Prediction error distribution
errors = y_test - rf_pred
plt.figure(figsize=(8,5))
sns.histplot(errors, kde=True)
plt.title("Random Forest Prediction Errors")
plt.xlabel("Error (actual - predicted)")
plt.show()

# Data Mining method : NEURAL NETWORK REGRESSION (MLPRegressor)

# Scale features for neural network
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Build MLPRegressor model
mlp = MLPRegressor(hidden_layer_sizes=(64,32), max_iter=500, random_state=123)
mlp.fit(X_train_scaled, y_train)

# Predict
mlp_pred = mlp.predict(X_test_scaled)

# Evaluate
mlp_mse = mean_squared_error(y_test, mlp_pred)
mlp_rmse = np.sqrt(mlp_mse)
mlp_mae = mean_absolute_error(y_test, mlp_pred)
mlp_r2 = r2_score(y_test, mlp_pred)
print("\nMLPRegressor Metrics:")
print(f" MAE : {mlp_mae:.3f}")
print(f" RMSE: {mlp_rmse:.3f}")
print(f" R2  : {mlp_r2:.4f}")

# Actual vs predicted
plt.figure(figsize=(8,6))
plt.scatter(y_test, mlp_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual cnt")
plt.ylabel("Predicted cnt")
plt.title("MLPRegressor: Actual vs Predicted")
plt.show()

# Error distribution
errors_mlp = y_test - mlp_pred
plt.figure(figsize=(8,5))
sns.histplot(errors_mlp, kde=True)
plt.title("MLPRegressor Prediction Errors")
plt.xlabel("Error (actual - predicted)")
plt.show()
```

```
print("\n\n Random Forest and Neural Network Regression Completed Successfully!")
```

OutPut:

Dataset Info:

```
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     731 non-null    int64
 1   dteday      731 non-null    object
 2   season      731 non-null    int64
 3   yr          731 non-null    int64
 4   mnth        731 non-null    int64
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
None
```

Summary Statistics:

```
          instant      season          yr        mnth     holiday     weekday  \
count  731.000000  731.000000  731.000000  731.000000  731.000000  731.000000
mean   366.000000    2.496580    0.500684    6.519836    0.028728    2.997264
std    211.165812    1.110807    0.500342    3.451913    0.167155    2.004787
min      1.000000    1.000000    0.000000    1.000000    0.000000    0.000000
25%    183.500000    2.000000    0.000000    4.000000    0.000000    1.000000
50%    366.000000    3.000000    1.000000    7.000000    0.000000    3.000000
75%    548.500000    3.000000    1.000000   10.000000    0.000000    5.000000
max    731.000000    4.000000    1.000000   12.000000    1.000000    6.000000

       workingday  weathersit        temp       atemp         hum   windspeed  \
count  731.000000  731.000000  731.000000  731.000000  731.000000  731.000000
mean     0.683995    1.395349    0.495385    0.474354    0.627894    0.190486
std      0.465233    0.544894    0.183051    0.162961    0.142429    0.077498
min      0.000000    1.000000    0.059130    0.079070    0.000000    0.022392
25%      0.000000    1.000000    0.337083    0.337842    0.520000    0.134950
50%      1.000000    1.000000    0.498333    0.486733    0.626667    0.180975
75%      1.000000    2.000000    0.655417    0.608602    0.730209    0.233214
max      1.000000    3.000000    0.861667    0.840896    0.972500    0.507463

            casual   registered          cnt
count   731.000000   731.000000   731.000000
mean    848.176471  3656.172367  4504.348837
std     686.622488  1560.256377  1937.211452
min       2.000000    20.000000    22.000000
25%     315.500000  2497.000000  3152.000000
50%     713.000000  3662.000000  4548.000000
75%    1096.000000  4776.500000  5956.000000
max    3410.000000  6946.000000  8714.000000
```

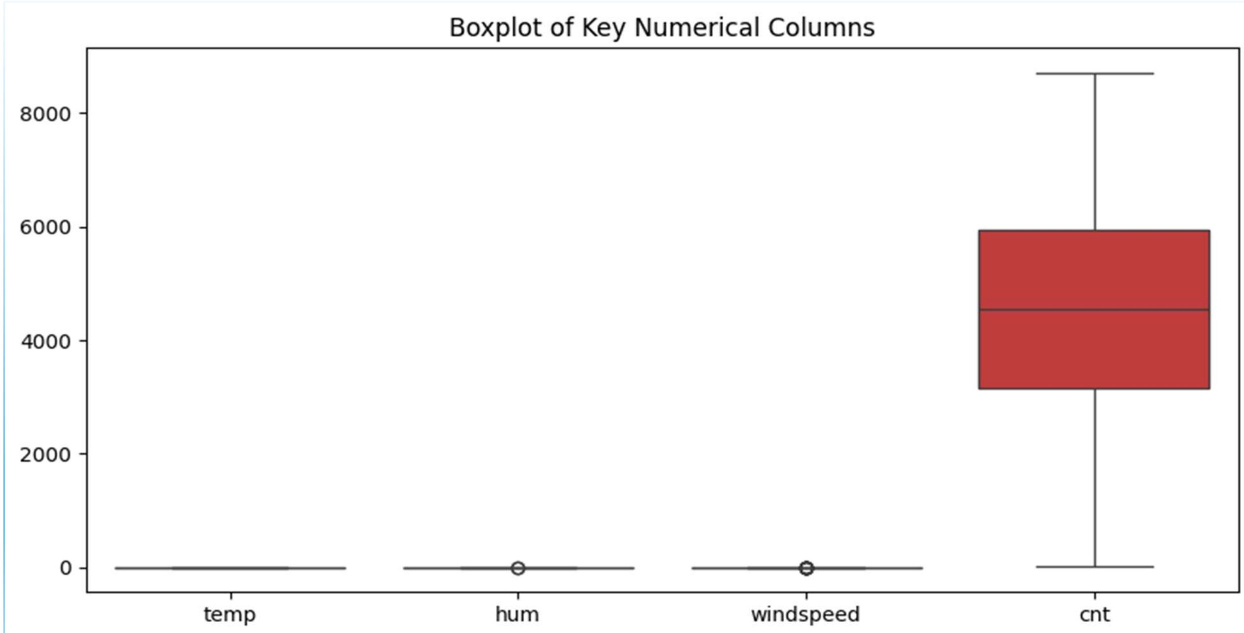
Missing Values:

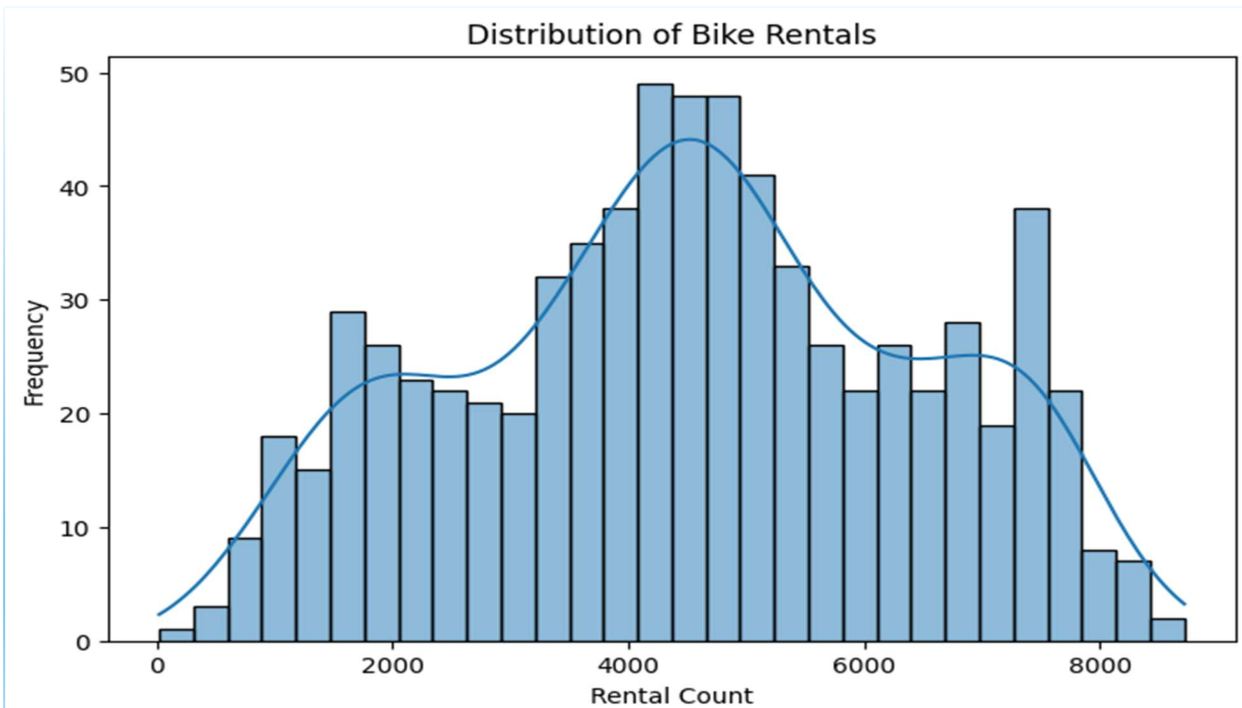
```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
```

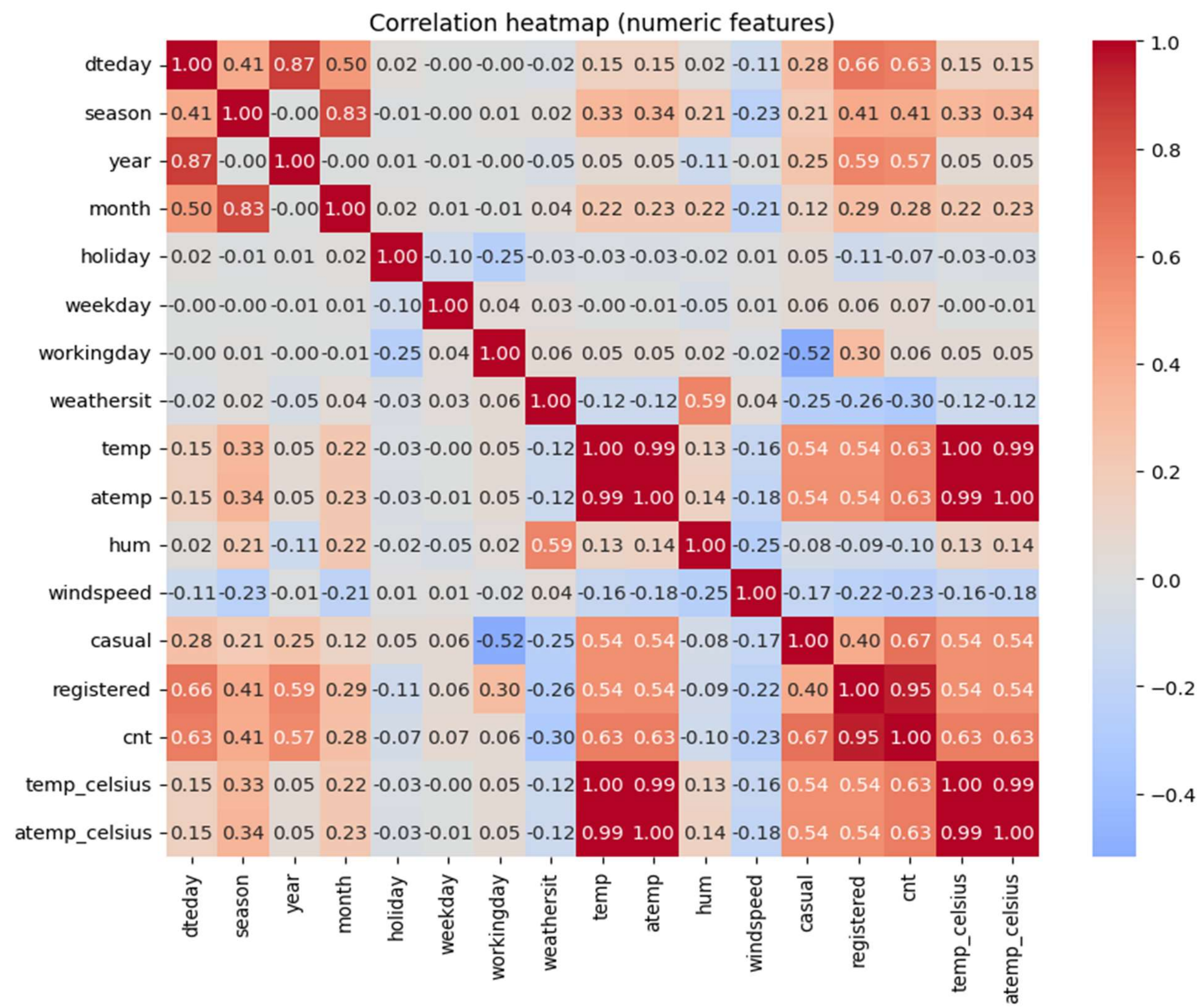
atemp 0
hum 0
windspeed 0
casual 0
registered 0
cnt 0
dtype: int64

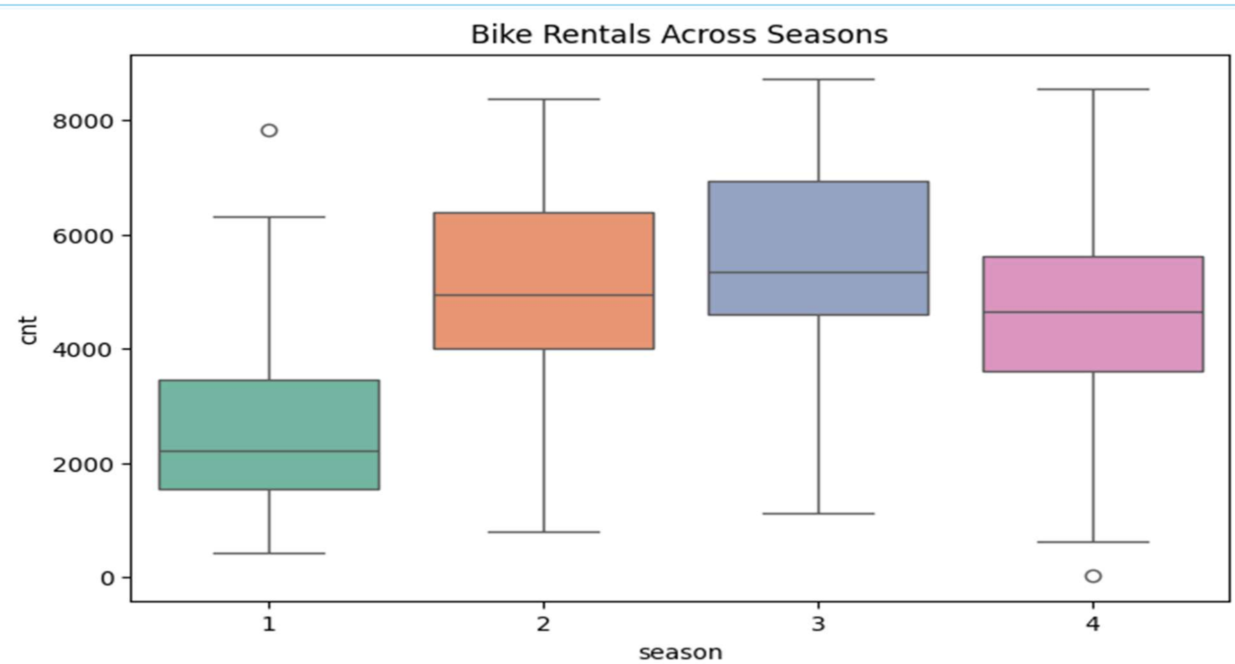
Dataset Shape: (731, 16)
Duplicate Rows: 0
Missing Values After Cleaning:
instant 0
dteday 0
season 0
yr 0
mnth 0
holiday 0
weekday 0
workingday 0
weathersit 0
temp 0
atemp 0
hum 0
windspeed 0
casual 0
registered 0
cnt 0
dtype: int64
Data Cleaning Completed.

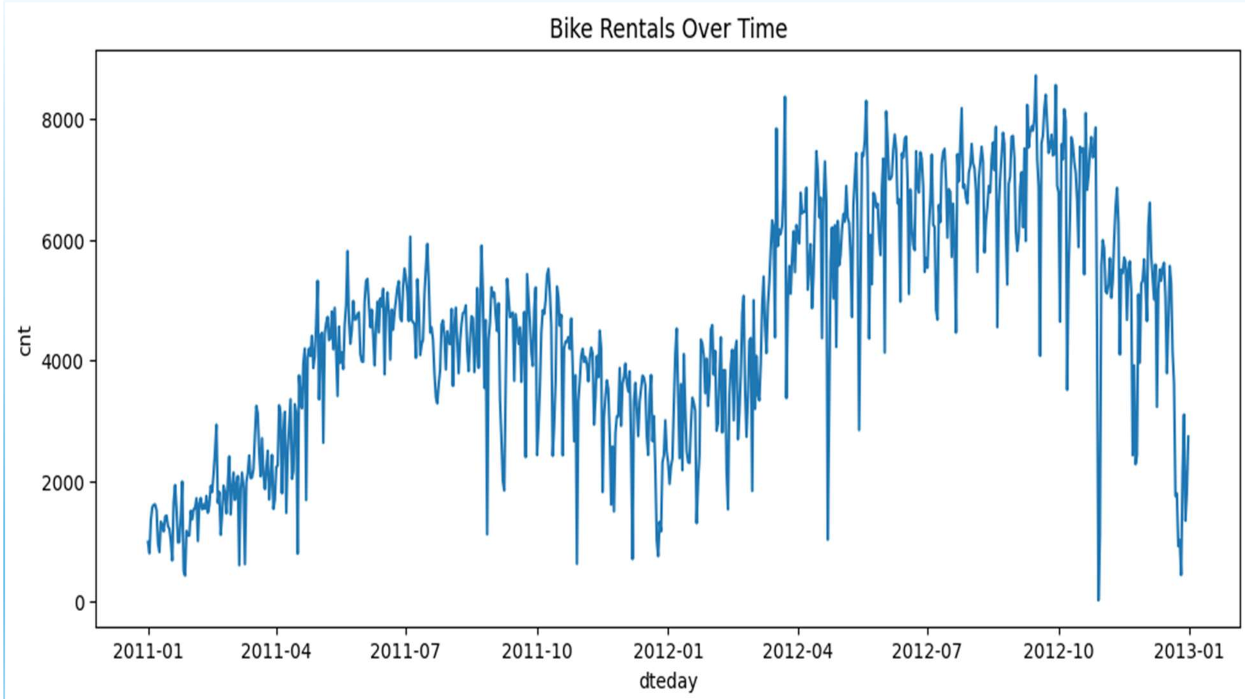
Skewness:
season -0.000384
year -0.002742
month -0.008149
holiday 5.654224
weekday 0.002742
workingday -0.793147
weathersit 0.957385
temp -0.054521
atemp -0.131088
hum -0.069783
windspeed 0.677345
casual 1.266454
registered 0.043659
cnt -0.047353
dtype: float64
Kurtosis:
season -1.342601
year -2.005487
month -1.209112
holiday 30.052462
weekday -1.254282
workingday -1.374686
weathersit -0.136467
temp -1.118864
atemp -0.985131
hum -0.064530
windspeed 0.410922
casual 1.322074
registered -0.713097
cnt -0.811922
dtype: float64


Boxplot of Key Numerical Columns
8000
6000
4000
2000
0
temp
hum
windspeed
cnt


Distribution of Bike Rentals
50
40
30
20
10
0
Frequency
0
2000
4000
6000
8000
Rental Count

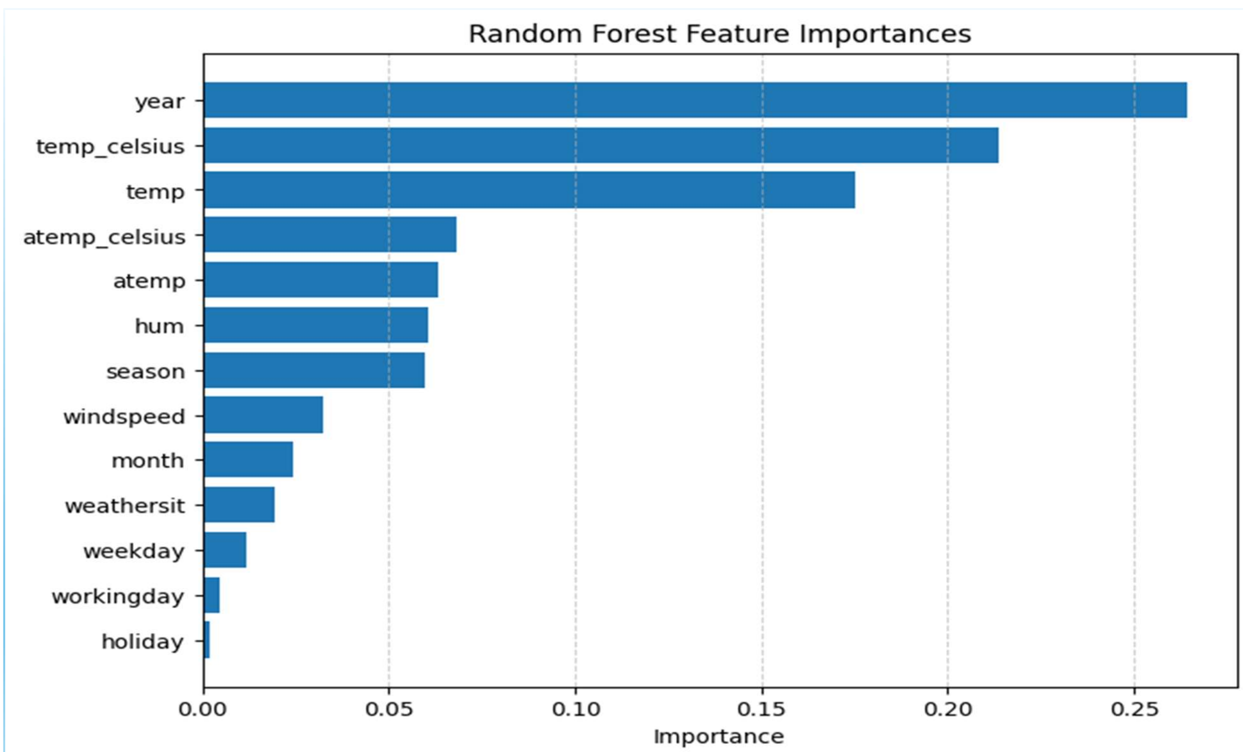

Correlation heatmap (numeric features)
dteday
season
year
month
holiday
weekday
workingday
weathersit
temp
atemp
hum
windspeed
casual
registered
cnt
temp_celsius
atemp_celsius
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

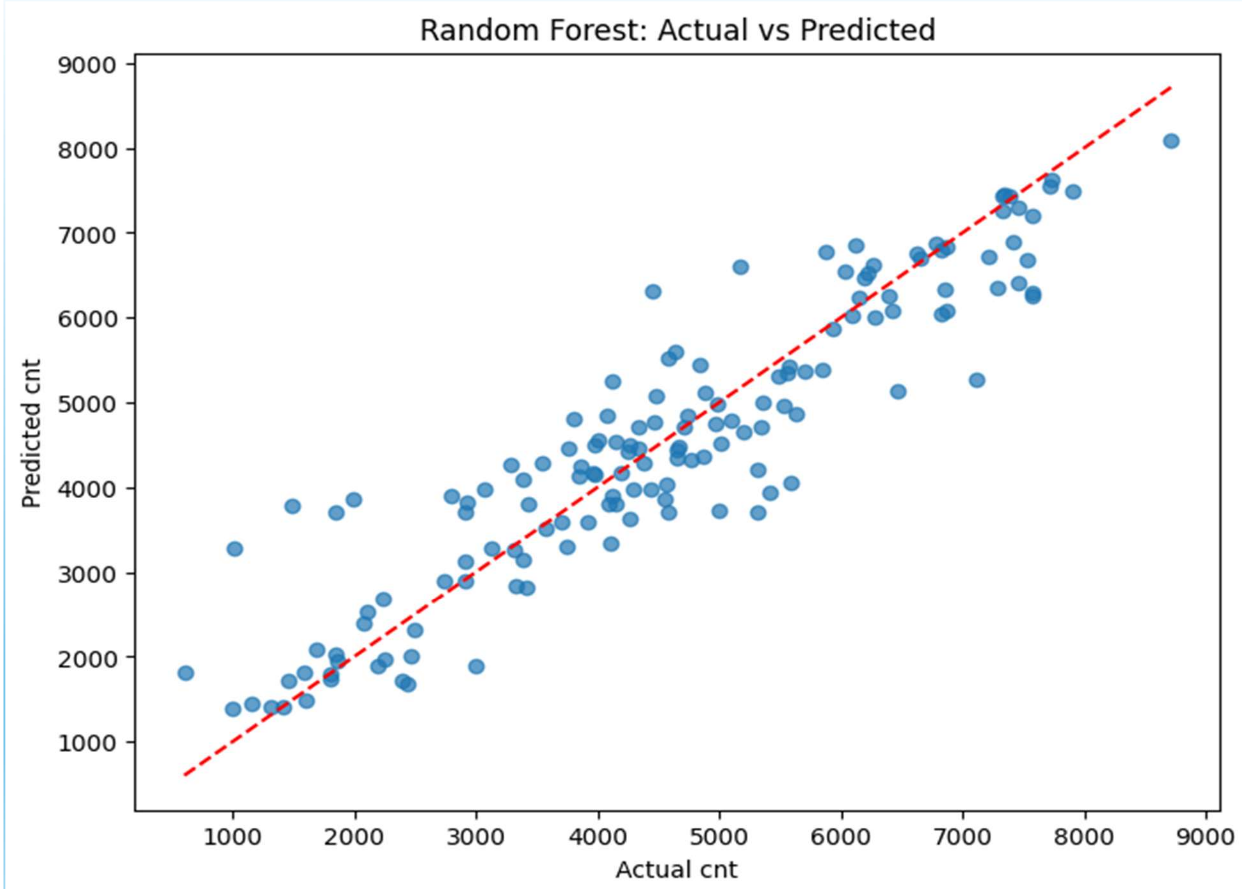

Bike Rentals Across Seasons
cnt
0
2000
4000
6000
8000
1
2
3
4
season

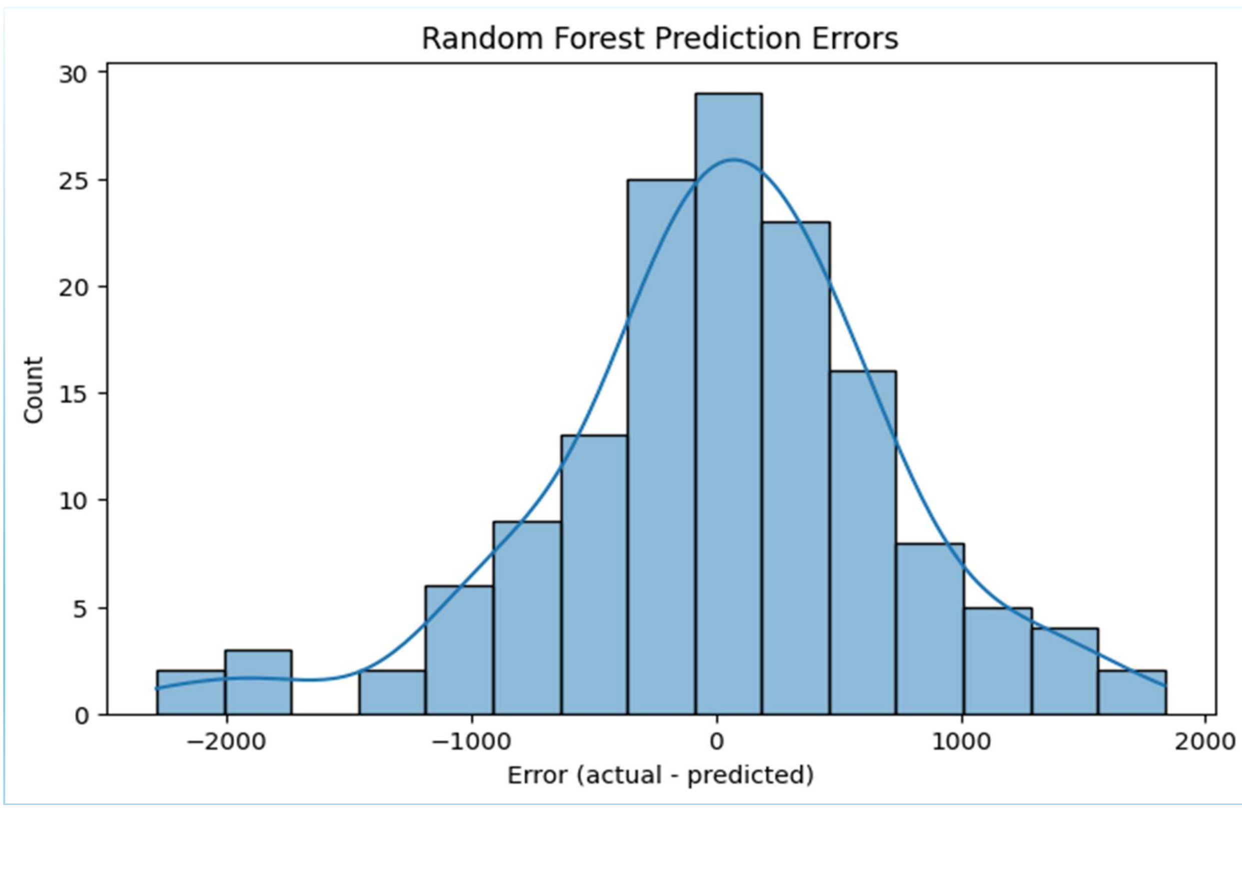


== =========== **Random Forest Regression** =====================

Random Forest Regression Metrics:
MAE : 525.838
RMSE: 711.411
R2 : 0.8517

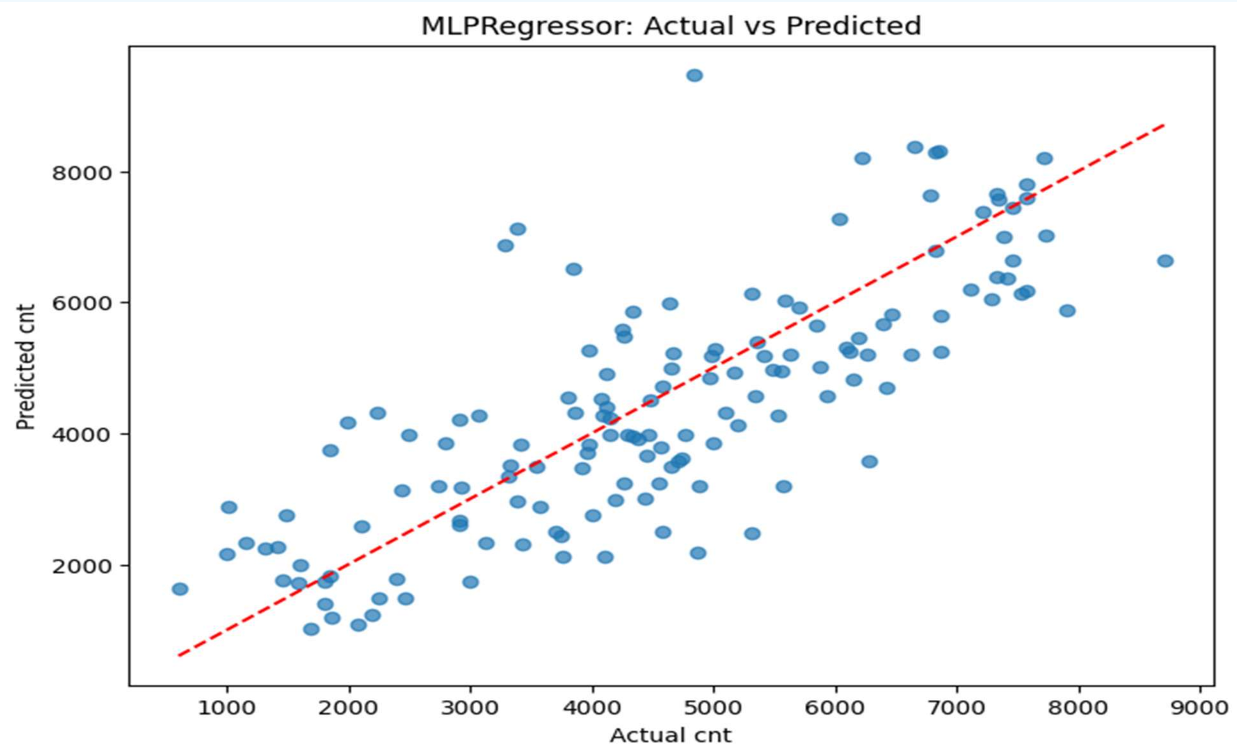


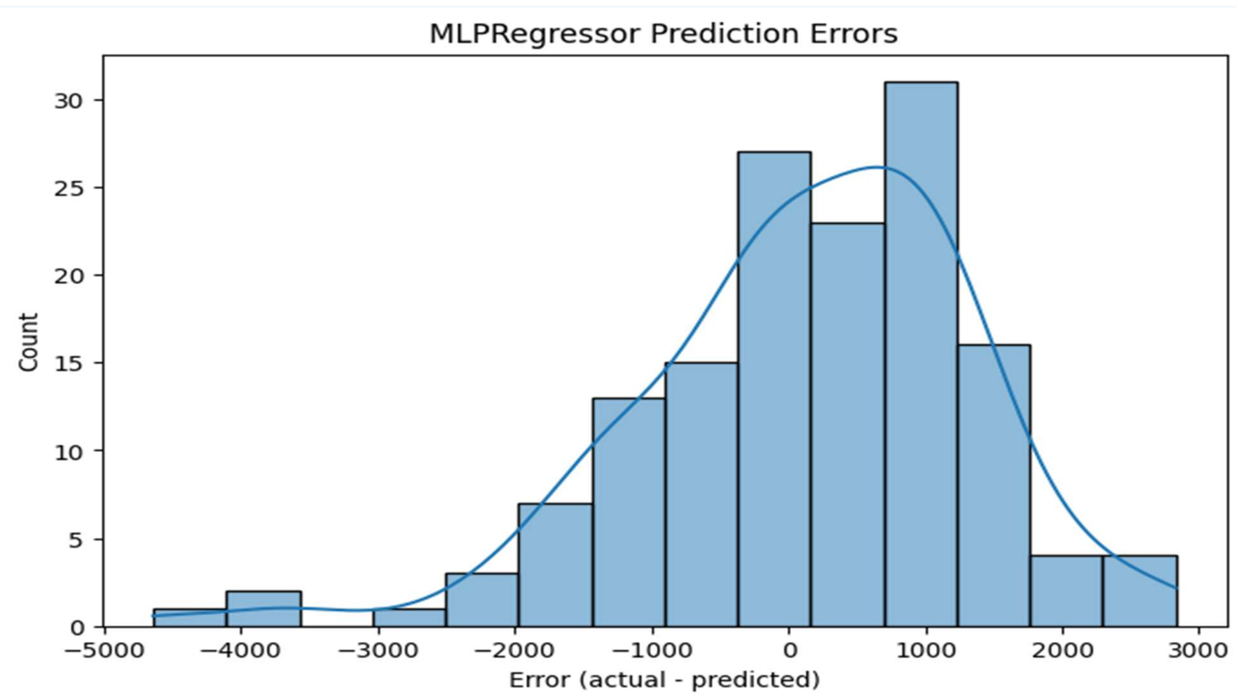

Random Forest: Actual vs Predicted
Predicted cnt
Actual cnt
1000
2000
3000
4000
5000
6000
7000
8000
9000


Random Forest Prediction Errors
Count
Error (actual - predicted)
0
5
10
15
20
25
30
−2000
−1000
0
1000
2000

================== Neural Network Regression (MLPRegressor) ============================

MLPRegressor Metrics:
MAE : 959.643
RMSE: 1230.695
R2 : 0.5561





Random Forest and Neural Network Regression Completed Successfully!