Capstone 3 Bike Sharing Dataset

Efraim William Solang - JCDSOL19

Table of Content

What you'll learn about today

ARE YOU READY?

Yes!

Of course!

Loading...

Tip: Select twice to customize this poll or quiz,
or go to **Elements** for more options.

Business Understanding

Bike Sharing Data

Every trip taken on a shared bike tells a story—and when collected at scale, these stories become powerful data.

Bike-sharing systems modernize traditional rentals by automating membership, pickup, and return. **Active in over 500 cities**, they support flexible mobility, environmental goals, and public health. Their detailed usage data enables researchers to **analyze urban travel patterns** and detect significant city events, acting as sensors in smart cities.

The Goals

In rapidly growing cities, traditional transport systems struggle to keep up with rising demands for efficiency, sustainability, and insight-driven planning. While bike-sharing systems offer eco-friendly mobility, their true value lies in the rich, structured data they generate—capturing trip durations, start and end points, and time of use. Yet, many organizations and city authorities fail to leverage this data effectively. The challenge is not just transportation—**it's uncovering patterns**, **predicting demand**, **and turning mobility data into actionable insights that drive smarter urban decisions.**

GOALS

01

Identify User Count Patterns

Discover trends and usage behavior from total bike-sharing user data.

02

Predict Users with Features

Build models using selected features to accurately forecast total user count.

03

Understand Machine Learning Algorithms

Learn how each algorithm processes data and makes predictions.

GOALS

04

Evaluate Tuned Model Performance

Analyze which model performs best after hyperparameter tuning.

05

Ensure Model Stakeholder Trust

Validate model reliability to ensure stakeholders can trust the predictions.

06

Conclusion & Recommendation

Create Conclusion and recommendation to improve the model.

Data Understanding

Total Data : 12.165 Data

- **dteday**: date
- **season**: season (1: winter, 2: spring, 3: summer, 4: fall)
- **hr**: hour (0 to 23)
- **holiday**: holiday or not
- **temp**: normalized temperature in Celsius. The values are derived via (t-tmin)/(tmax-tmin), tmin=-8, t_max=+39 (only in hourly scale)
- **atemp**: Normalized feeling temperature in Celsius. The values are derived via (t-tmin)/(tmax-tmin), tmin=-16, t_max=+50 (only in hourly scale)
- **hum**: normalized humidity. The values are divided into 100 (max)
- **casual**: count of casual users
- **registered**: count of registered users
- **cnt**: count of total rental bikes including both casual and registered
- **weathersit**

1: Clear, Few clouds, Partly cloudy, Partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

```
df.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12165 entries, 0 to 12164
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   dteday      12165 non-null  object 
 1   hum         12165 non-null  float64
 2   weathersit  12165 non-null  int64  
 3   holiday     12165 non-null  int64  
 4   season      12165 non-null  int64  
 5   atemp       12165 non-null  float64
 6   temp        12165 non-null  float64
 7   hr          12165 non-null  int64  
 8   casual      12165 non-null  int64  
 9   registered  12165 non-null  int64  
 10  cnt         12165 non-null  int64  
dtypes: float64(3), int64(7), object(1)
memory usage: 1.0+ MB
```

Preprocessing Data

The process of transforming raw data into a usable format for analysis or machine learning models

It involves cleaning, transforming, and integrating data to improve its quality and make it more suitable for downstream tasks.

Split date to each Month and Day name

```
from datetime import datetime


df['year'] = pd.DatetimeIndex(df['dteday']).year
df['month'] = df['dteday'].dt.month_name()
df['day'] = df['dteday'].dt.day_name()

month_order = ["January", "February", "March", "April", "May", "June",
               "July", "August", "September", "October", "November", "December"]

df['month'] = pd.Categorical(df['month'], categories=month_order, ordered=True)
```

✓ 0.0s

Day Category

```
weekend_days = ["Saturday", "Sunday"]

df['week_category'] = df['day'].apply(
    lambda x: 'Weekend' if x in weekend_days else 'Weekday'
)
```

Count Temperature by celcius

```
df['temp_celsius'] = df['temp']  * (39-(-8)) + (-8)

df.drop('temp', axis=1, inplace=True)
```

Hour Category

```
hour_category = [0, 6, 12, 18, 24]
hour_labels = ['Early Morning', 'Morning', 'Afternoon', 'Evening']
df['hour_category'] = pd.cut(df['hr'], bins=hour_category, labels=hour_labels, right=False)
```

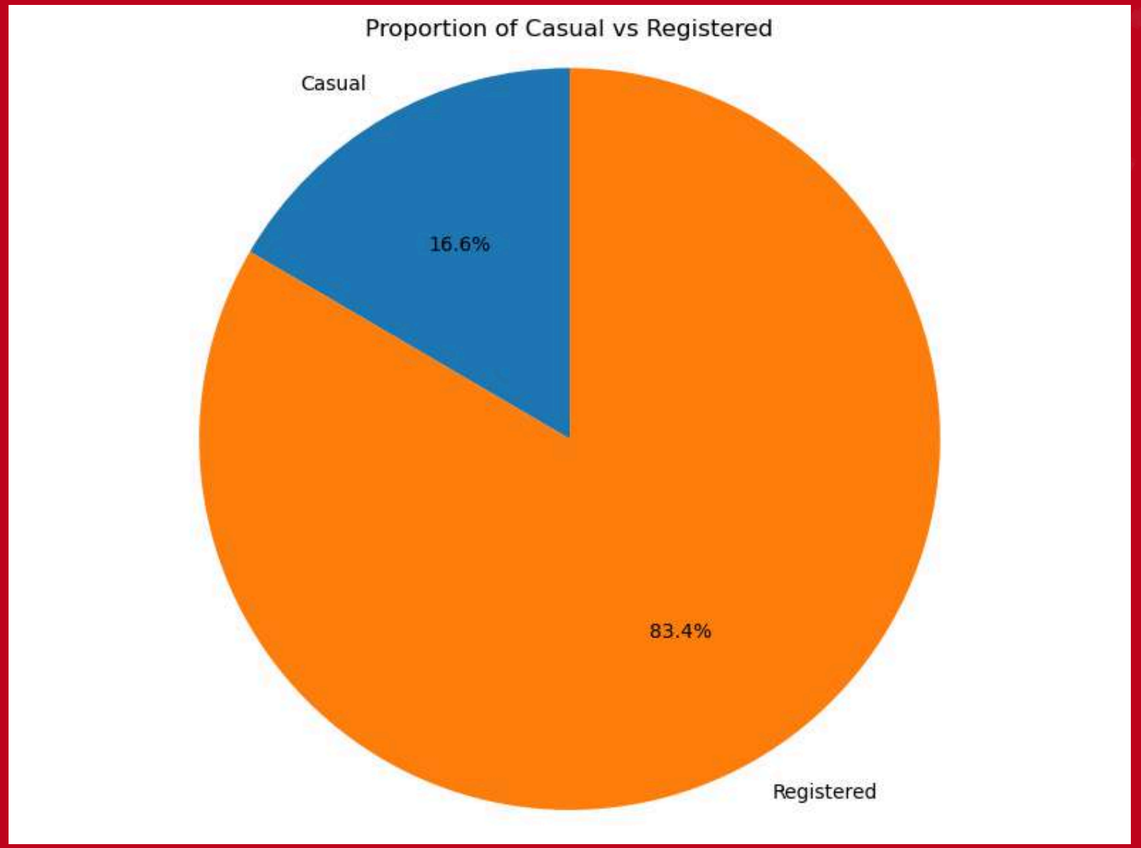
Exploratory Data Analysis

Proportion

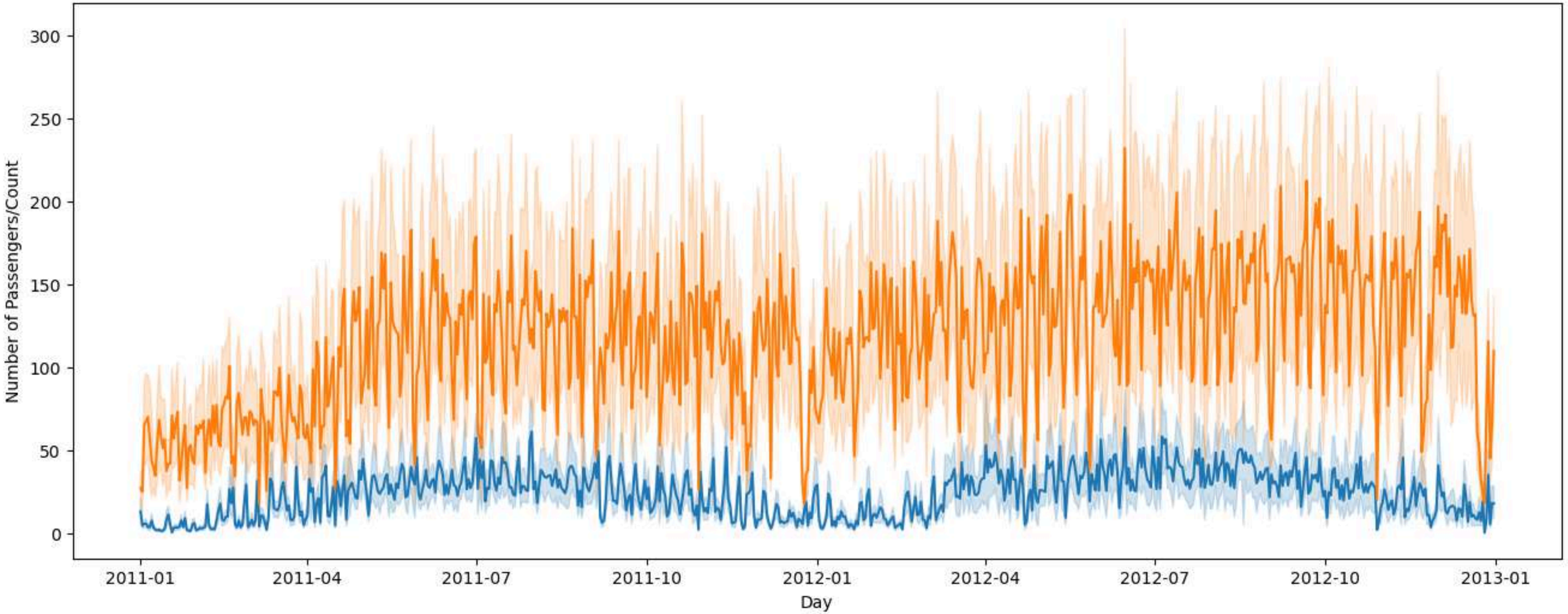
of Casual vs Registered

The most of user are Registered with 83.4%

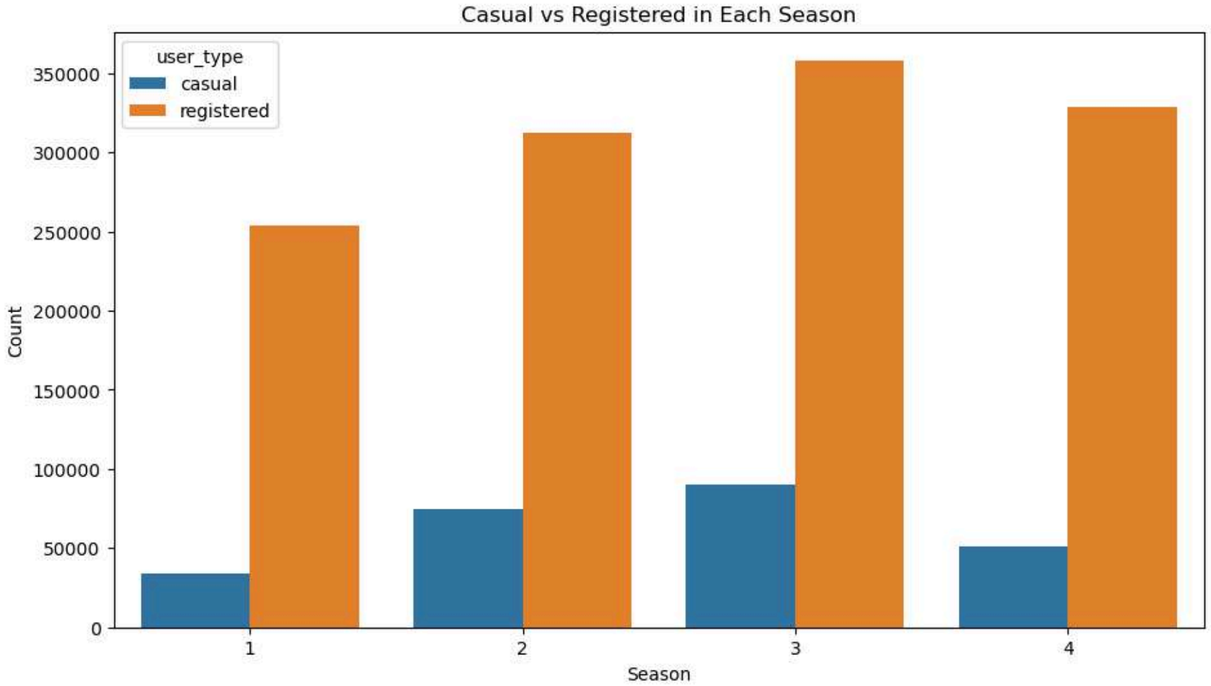
Data shows that more people use Registered bicycles than Casual ones.



Count Time Series

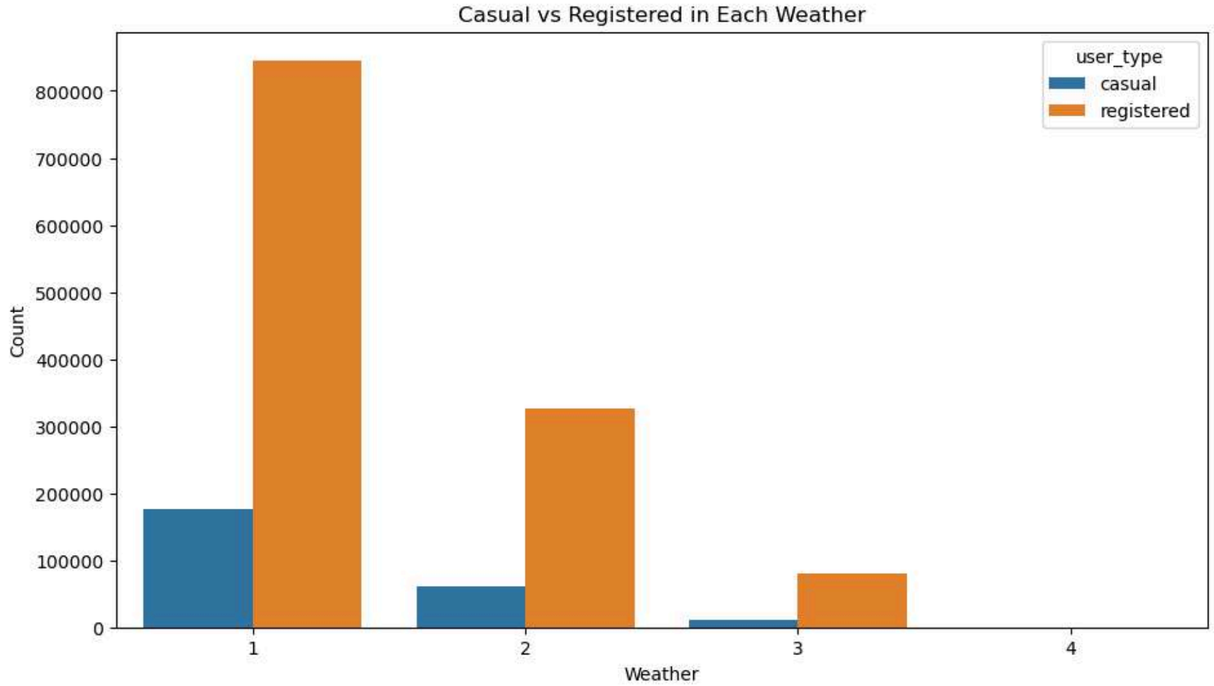


User Each Season and Weather



The graph shows that the largest number of bicycle users, both Casual and Registered, occurs in the **Summer**.

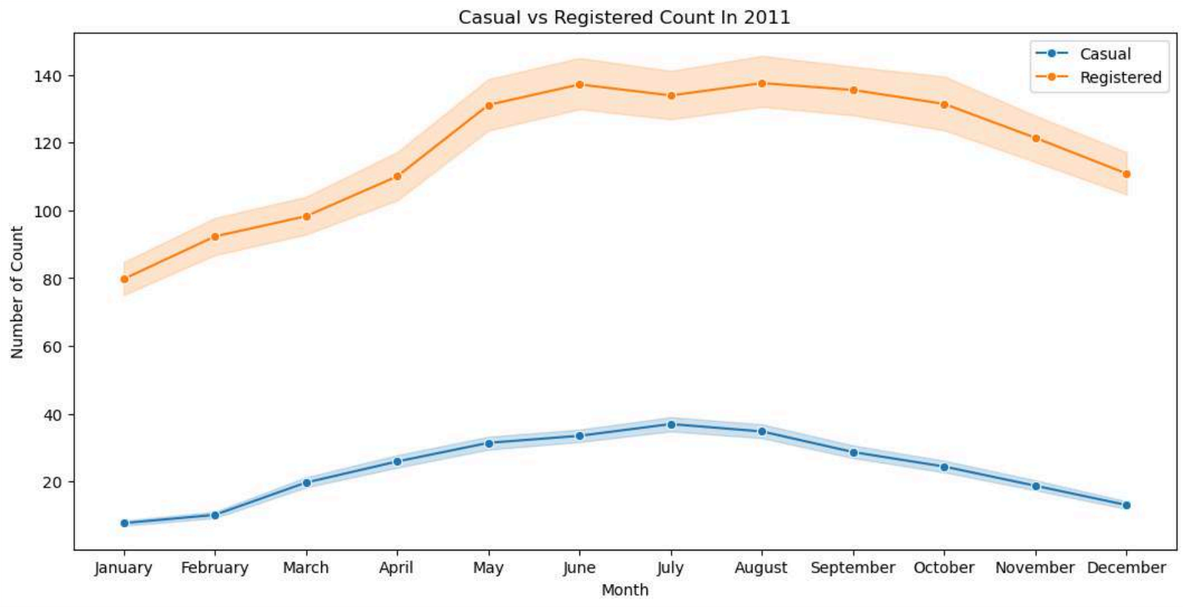
Notes :
1: winter
2: spring
3: summer
4: fall

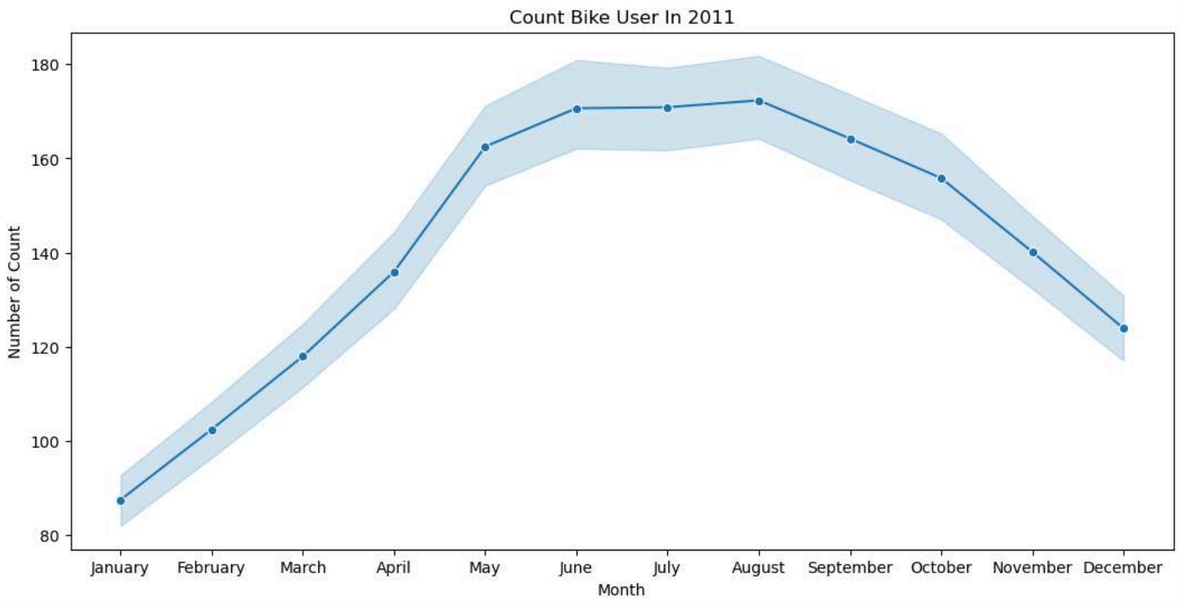


The graph shows that the most popular weather for cycling is **sunny or cloudy weather**.

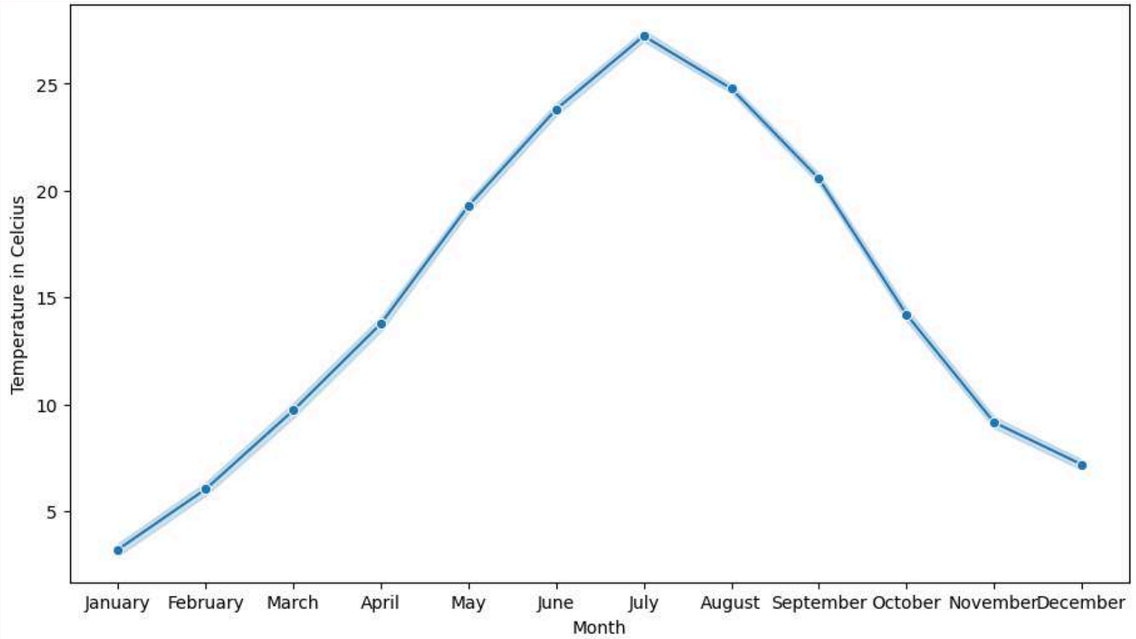
Notes :
- 1: Clear, Few clouds, Partly cloudy, Partly cloudy
- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

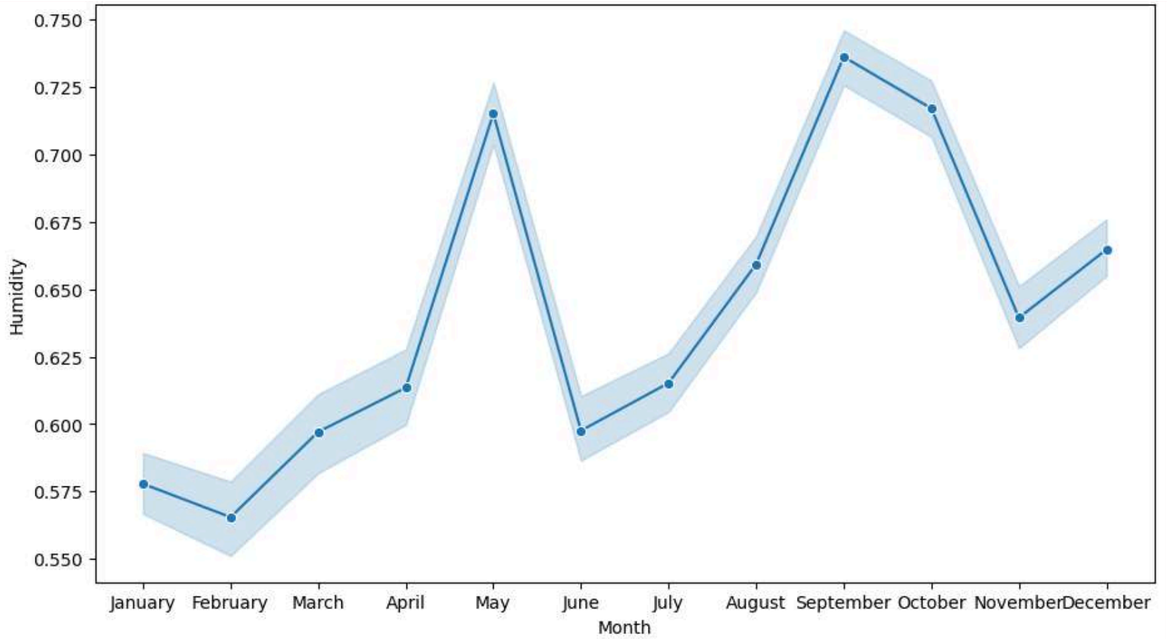
Count per Month in 2011



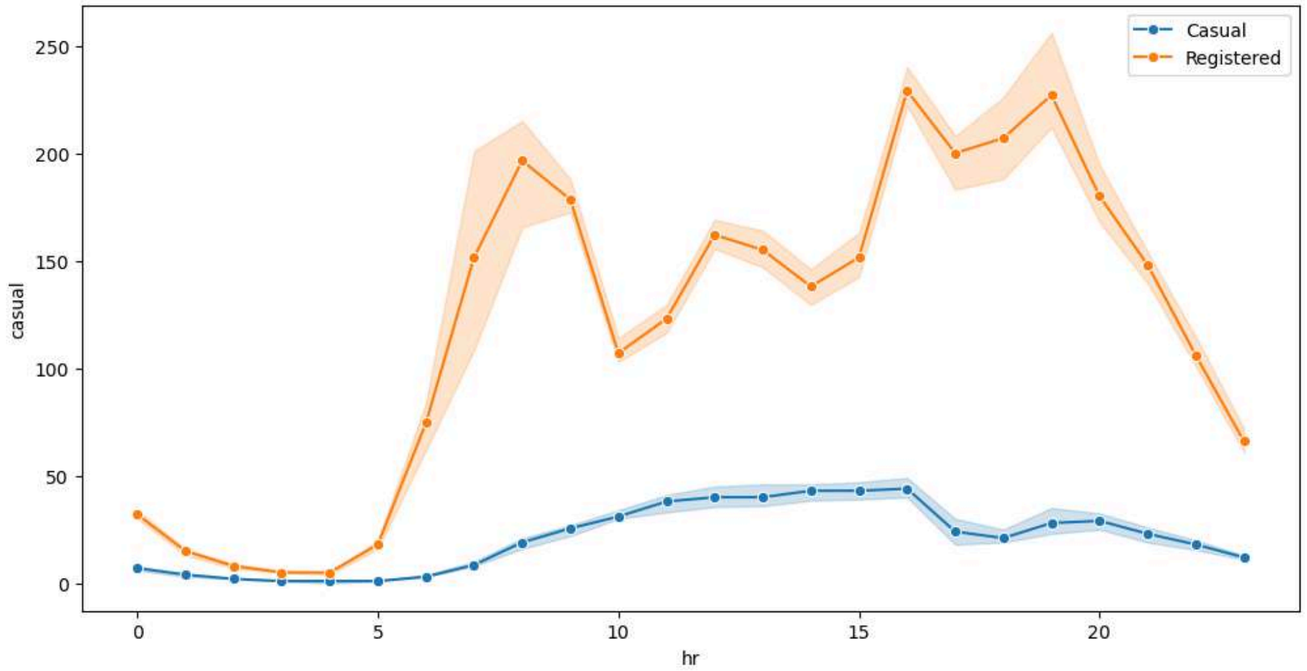


Tempearture & Humidity Each Month





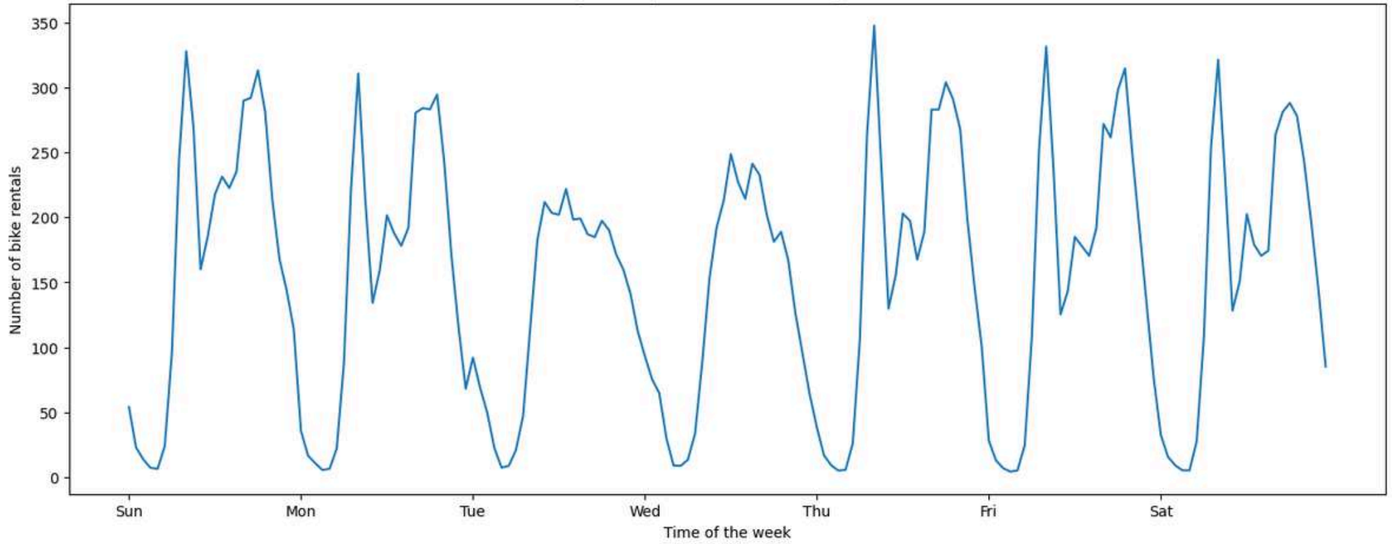
Count per Hour



The data shows that there is an increase in users starting from **6 AM to 9 AM** and the peak is at **4 PM**. Then at midnight there is a decrease.

Average Hour Bike Demand during the Week

The graph on the side shows that **working hours (**especialy during **Morning)** are the hours when bicycles are most frequently used.

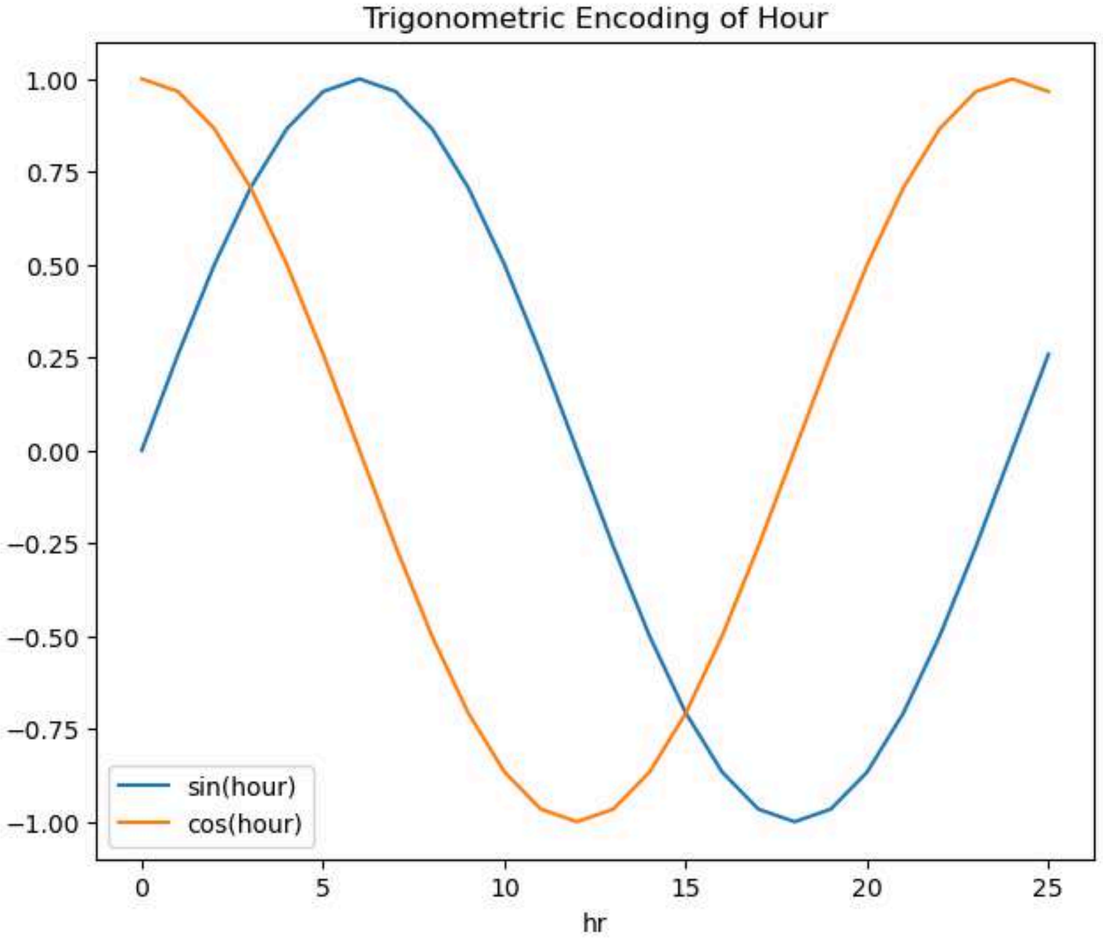


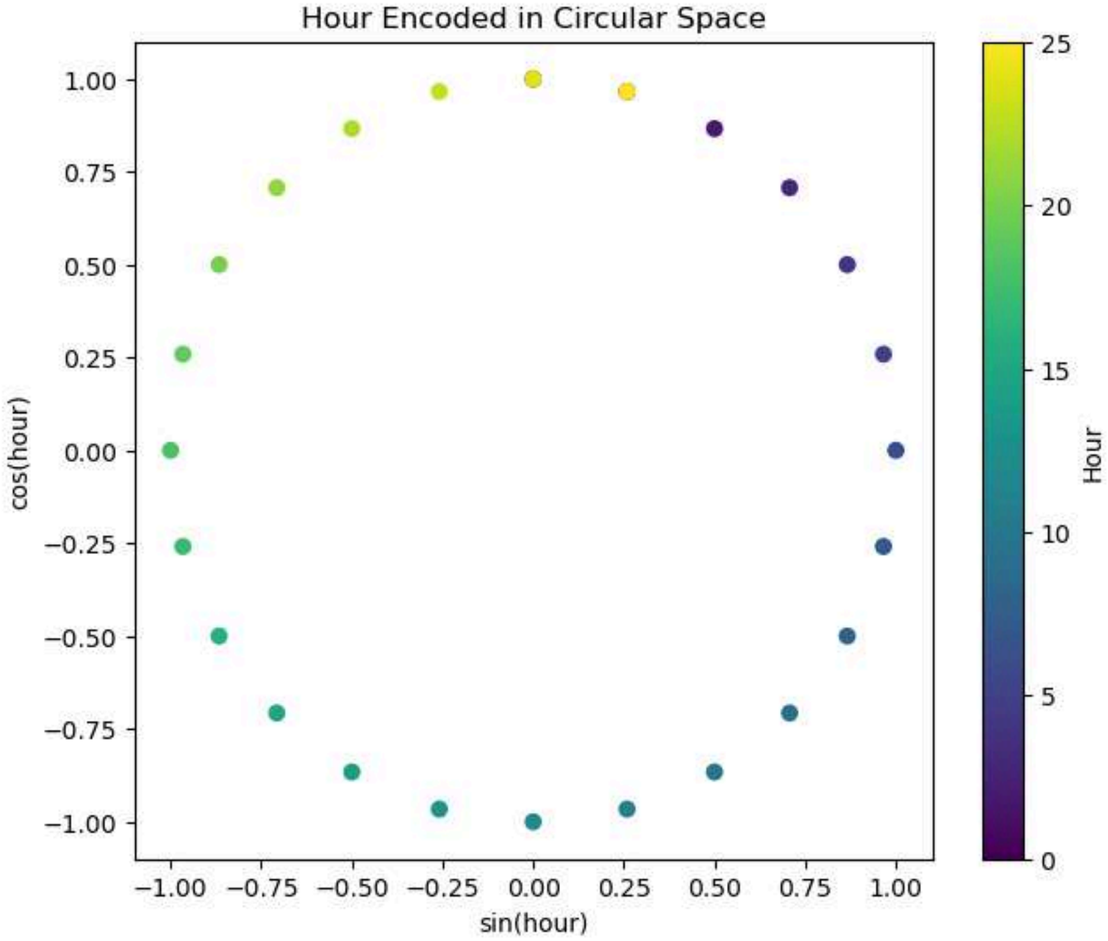
Feature Engineering

The process of transforming raw data into features that are **more suitable** for machine learning models

It involves selecting, transforming, and creating new features to improve model performance.

Trigonometric Encoding Hour





Sin & Cos Transformer

```python
from sklearn.preprocessing import FunctionTransformer

def sin_transformer(period):
    return FunctionTransformer(lambda x: np.sin(x / period * 2 * np.pi))

def cos_transformer(period):
    return FunctionTransformer(lambda x: np.cos(x / period * 2 * np.pi))
```

Hour Trigonometric Encoding

```python
hour_df = pd.DataFrame(np.arange(26).reshape(-1, 1), columns=["hr"])

hour_df["hour_sin"] = sin_transformer(24).fit_transform(hour_df)["hr"]
hour_df["hour_cos"] = cos_transformer(24).fit_transform(hour_df)["hr"]
```

Week Trigonometric Encoding

```python
# Week Trigonometric Encoding
dayof_week_names = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']

day_of_week_df = pd.DataFrame(np.arange(7).reshape(-1, 1), columns=["day"])

day_of_week_df["day_of_week_sin"] = sin_transformer(7).fit_transform(day_of_week_df)["day"]
day_of_week_df["day_of_week_cos"] = cos_transformer(7).fit_transform(day_of_week_df)["day"]
day_of_week_df["day"] = dayof_week_names

df = df.merge(day_of_week_df[['day', 'day_of_week_sin', 'day_of_week_cos']], on='day', how='left')
```

Month Trigonometric Encoding

```
# Month Trigonometric Encoding
month_names = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']

month_df = pd.DataFrame(np.arange(12).reshape(-1, 1), columns=["month"])

month_df["month_sin"] = sin_transformer(12).fit_transform(month_df)["month"]
month_df["month_cos"] = cos_transformer(12).fit_transform(month_df)["month"]
month_df["month"] = month_names

df = df.merge(month_df[['month', 'month_sin', 'month_cos']], on='month', how='left')
```

Year Trigonometric Encoding

```
# Year Trigonometric Encoding
year_names = ['2011', '2012']

year_df = pd.DataFrame(np.arange(2).reshape(-1, 1), columns=["year"])

year_df["year_sin"] = sin_transformer(2).fit_transform(year_df)["year"]
year_df["year_cos"] = cos_transformer(2).fit_transform(year_df)["year"]
year_df["year"] = year_names

df['year'] = df['year'].astype(str)
df = df.merge(year_df[['year', 'year_sin', 'year_cos']], on='year', how='left')
```

```
df.columns
```
✓ 0.0s

```
Index(['dteday', 'hum', 'weathersit', 'holiday', 'season', 'hr', 'casual',
       'registered', 'cnt', 'year', 'month', 'day', 'temp_celsius',
       'week_category', 'hour_category', 'hour_sin', 'hour_cos',
       'day_of_week_sin', 'day_of_week_cos', 'month_sin', 'month_cos',
       'year_sin', 'year_cos'],
      dtype='object')
```

```
X = df.drop(['dteday','hr', 'casual', 'registered', 'cnt', 'year', 'month', 'day'], axis=1).copy()

# Log Trasnform
y = np.log1p(df['cnt']).copy()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2042)
```

```
categorical_cols = ['weathersit', 'holiday', 'season', 'hour_category', 'week_category']
numerical_cols = ['temp_celsius']

encoding = ColumnTransformer([
    ('One Hot Encoding', OneHotEncoder(drop='first'), categorical_cols),
    ('Standart Scaller', StandardScaler(), numerical_cols )
], remainder='passthrough')
```

Feature Selection & Split

The process of **selecting** a subset of relevant features (variables, predictors) from a larger set to use in model construction

This process aims to improve model performance, reduce computational cost, and enhance interpretability by focusing on the most informative attributes.

METRICS

01

Mean Squared Error

Mean squared error (MSE) measures the amount of error in statistical models. It assesses the average squared difference between the observed and predicted values.

02

Root Mean Squared Error

The root mean square error (RMSE) measures the average difference between a statistical model's predicted values and the actual values

03

R-Squared

R-squared (R^2) is defined as a number that tells you how well the independent variable(s) in a statistical model explains the variation in the dependent variable. It ranges from 0 to 1, where 1 indicates a perfect fit of the model to the data.

METRICS

04

MAPE

Mean absolute percentage error measures the average magnitude of error produced by a model, or how far off predictions are on average.

METRICS & BUSINESS

MSE & RMSE

Determining the operational margin of error — for example: are predictions accurate enough to determine the amount of stock, vehicles, labor, etc.

R-Squared

Measures confidence in the model — the closer it is to 1, the more confident management is that prediction-based decisions are appropriate.

MAPE

Suitable for business communications because it is in percent — helps answer: "how big is the average prediction error from the target?"

METRICS & BUSINESS EXAMPLE

RMSE = 30 → there is a possibility of a **difference** of ±30 vehicles, can help manage stock.

MAPE = 23% → the average prediction **is 23% off**, so management can take a buffer or cost compensation if the prediction is used for supply estimation.

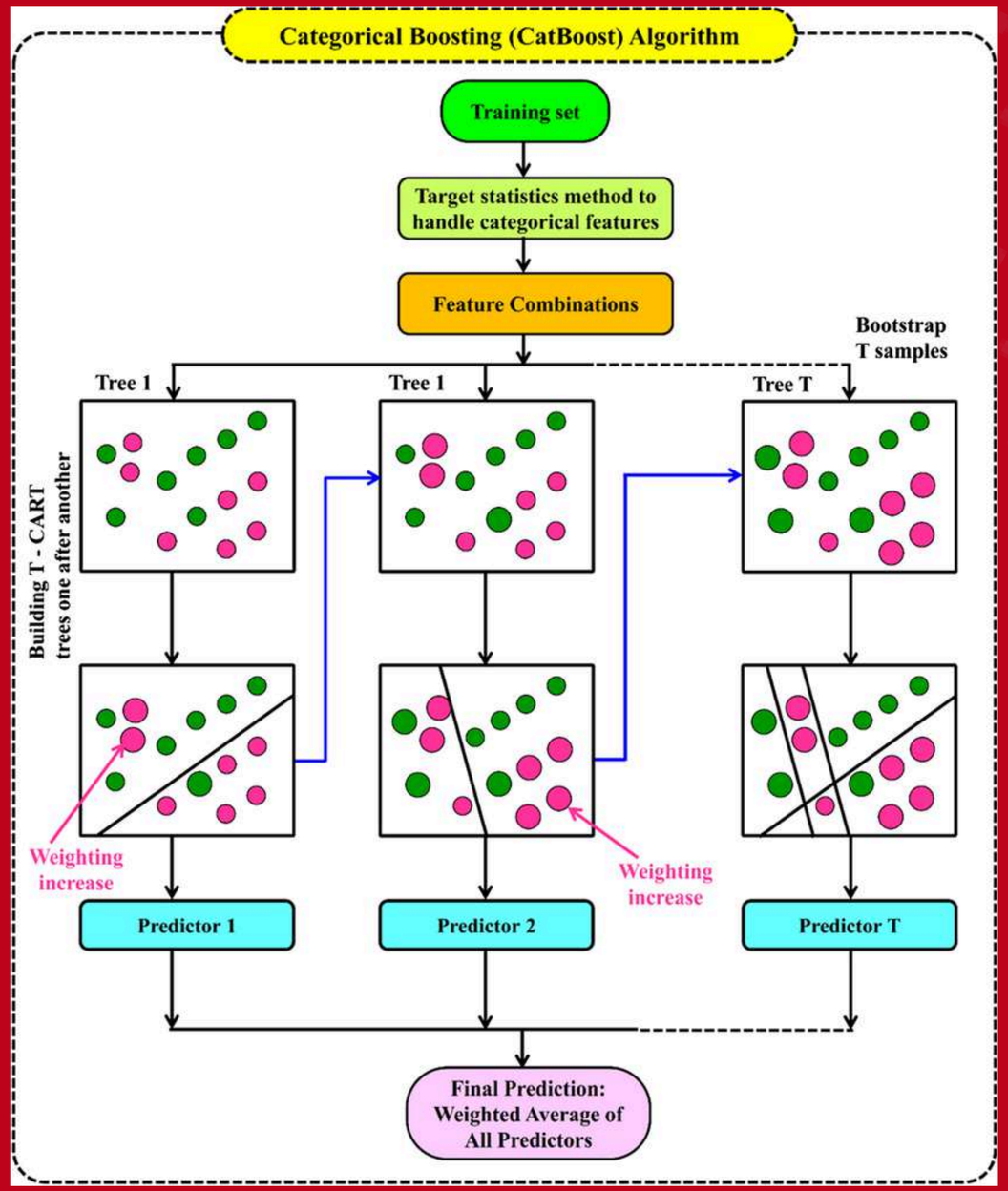
R^2 = 0.93 → management can **trust** that the model has explained the data pattern well, and can be used as a basis for decision making.

Modeling

	Model	MAE	RMSE	R2	MAPE
0	XGB_model	0.2264	0.3244	0.9424	0.0772
1	CAT_model	0.2107	0.3044	0.9493	0.0724
2	LR_model	0.5234	0.6848	0.7433	0.1734
3	KN_model	0.4184	0.5646	0.8255	0.1391
4	DTR_model	0.3341	0.4937	0.8665	0.1099
5	SVR_model	0.2985	0.4149	0.9057	0.1021

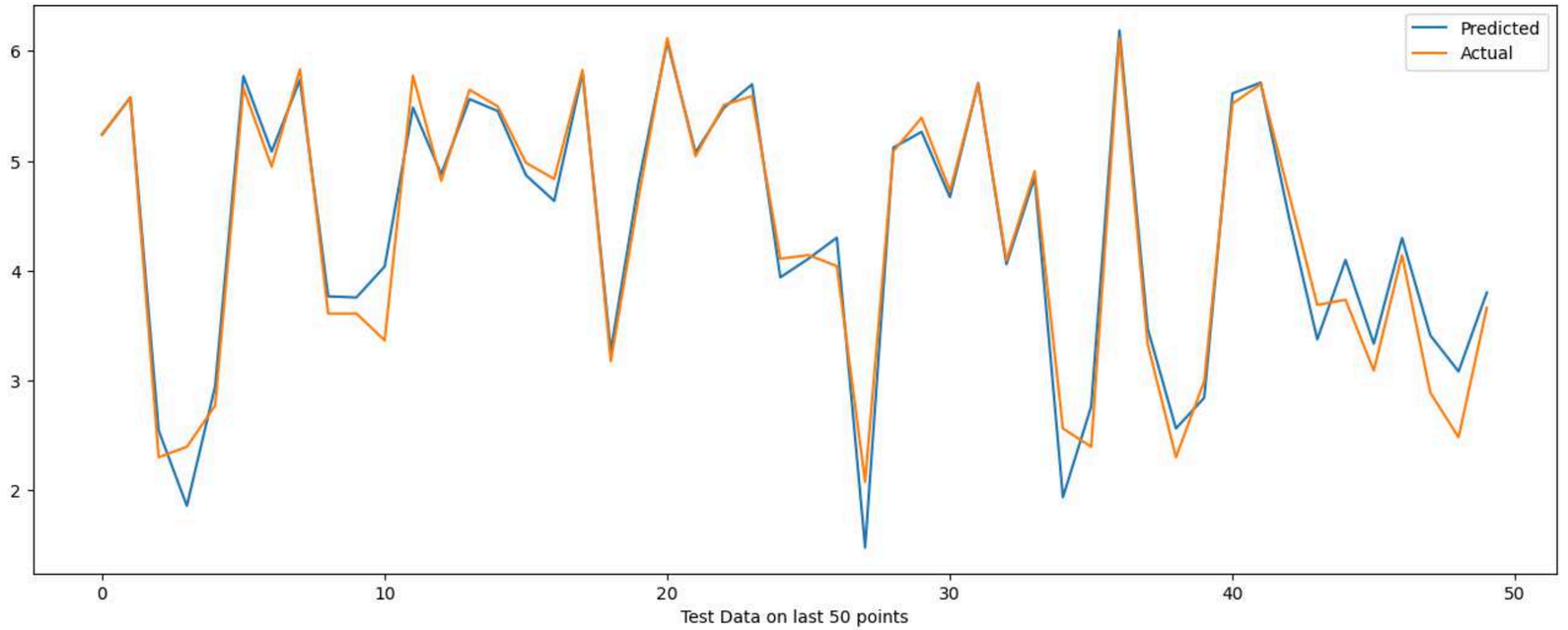
CatBoost Regression

CatBoost is an advanced gradient-boosting library specifically designed to address the challenges of **handling categorical data in machine learning**. In contrast to other gradient boosting techniques, CatBoost is a superior option for tasks **involving complicated**, real-world datasets since it is good at **handling categorical** information natively.



Prediction to Test Set

...	# MAE	# RMSE	# R2	# MAPE
Without Tunning	19.9716	32.3896	0.9287	0.2447
Grid Tunning	20.0644	32.5511	0.9279	0.2448
Random Tunning	18.7655	30.4804	0.9368	0.2352



Based on this graph, there are three comparisons in the assessment. Namely the CATBoostRegression model without Tunning, with GridSearchCV Tunning, and with RandomForestCV Tunning. **The RandomSearchCV value shows the best results.**

Best Parameter (RandomSearchCV)

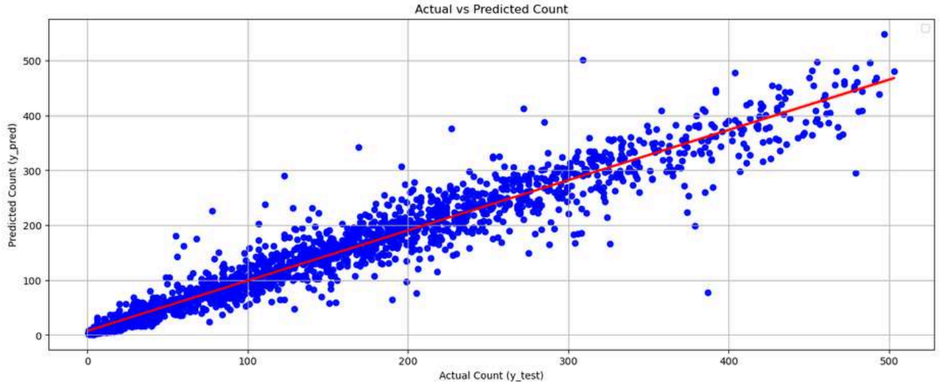
```python
param_grid = {
    "iterations": [200, 500, 700],
    "depth": [4, 6, 8, 10],
    "learning_rate": [0.01, 0.05, 0.1],
    "l2_leaf_reg": [1, 3, 5, 7, 9],
    "bagging_temperature": [0, 1, 3, 5],
    "random_strength": [1, 5, 10],
    "border_count": [32, 64, 128]
}
```

```
CATBoost
Best_score: -0.30141209641540206
Best_params: {'random_strength': 10, 'learning_rate': 0.1, 'l2_leaf_reg': 5, 'iterations': 500, 'depth': 10, 'border_count': 128, 'bagging_temperature': 1}
```

Random Tunning	18.7655	30.4804	0.9368	0.2352

The CatBoost model shows good performance with the best parameters generated by the model itself. The parameters are able to **capture diverse patterns**, **avoid overfitting**, and are sensitive to **numerical & categorical features**, resulting in accurate and stable predictions.

Model Limitation



	Range	Score MAE	Score MAPE	Score RMSE	Score R2
0	<=50	5.220548934160844	0.4148556951146379	70.38543469619954	0.7424154701802148
1	51-100	17.676206935201172	0.2198859011524058 3	897.0547223248774	0.17929476458619353
2	101-150	20.902953337975493	0.1720482385487989	860.3573821377602	0.22039807580996051
3	151-200	24.596861519902763	0.14134231160864807	1327.7315968895305	0.15036354397520957
4	201-250	26.635824520208185	0.12422214891420591	1402.081432740753	0.0852973776959135
5	251-300	32.50785618803916	0.11984937126602938	1842.7119230512494	0.11914127437958633
6	301-350	34.78911900987872	0.10414056172048627	2239.271596030173	0.09823640436313297
7	351-400	40.15213325642647	0.10528135659269695	2650.828504654054	-0.036279760080113466
8	401-450	28.87054760290319	0.06804986980363563	1282.6343076712367	0.06136836037166138
9	451-500	20.153889149662188	0.04411295775698867	604.4807322264767	0.10145704651040455
10	>501	51.14943040687052	0.10291635896754633	2616.264230947291	*Missing value*
11	All Count Range (Max 548)	18.76551469365117	0.2351921383722612	929.0567425689343	0.936817474135186

The model **performs very well at low-value predictions** (≤50), with an MAE of only 5.22 and a **high R^2 of 0.74**, indicating that the model is **very accurate for this range**.

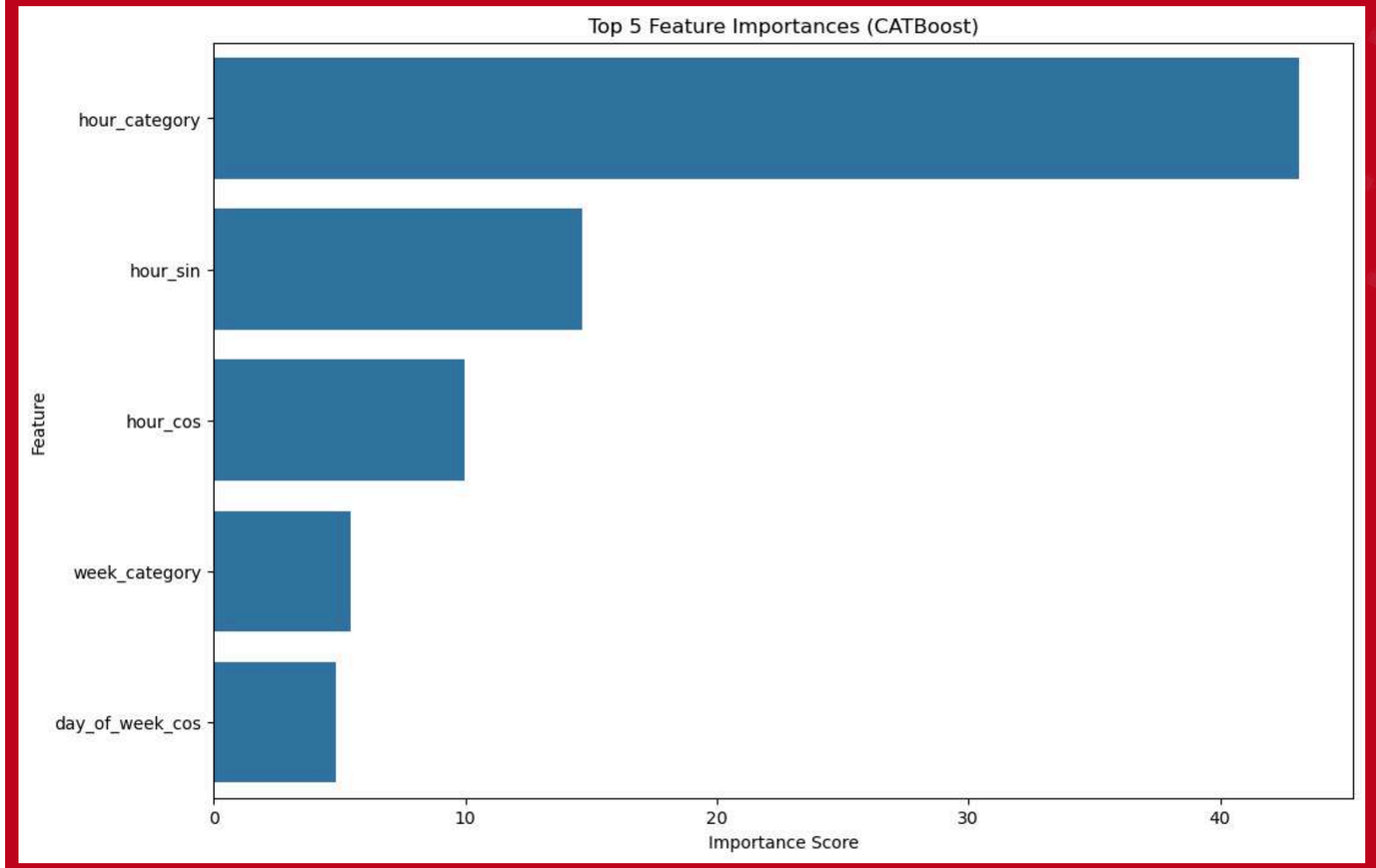
However, as the prediction increases (151–400), the MAE and RMSE increase sharply, and the R^2 even decreases to negative values, such as in the range 351–400, indicating that the prediction is **worse than the average guess**.

Overall, the model **performs very well in small-medium cases**, but **needs to improve generalization for large-value predictions.**

Feature Importances

Techniques used in machine learning to **determine the relevance** of each input feature in predicting a target variable

Feature Importance which is very influential is **'hour_category'**, **'hour_sin'**, **'hour_cos'**, **'week_category'**, **'day_of_week_cos'**.



Conclusion

The best model is CATBoost

The regression model showed excellent performance with an R^2 of 0.9368, meaning that about 94% of the target variation was successfully explained by the model. The MAE value of 18.77 and RMSE of 30.48 indicated that the average model prediction error was quite low and stable. In addition, the MAPE of 23.52% indicated that in terms of percentage, the model prediction was relatively accurate. Overall, the model has **high accuracy** and **good generalization to the test data**.

```
--- Regression Model Metrics ---
Mean Absolute Error (MAE)                       : 18.77
Root Mean Squared Error (RMSE)                  : 30.48
R-squared (R2 Score)                            : 0.9368
Mean Absolute Percentage Error (MAPE)           : 0.2352
------------------------------------------------------------
```

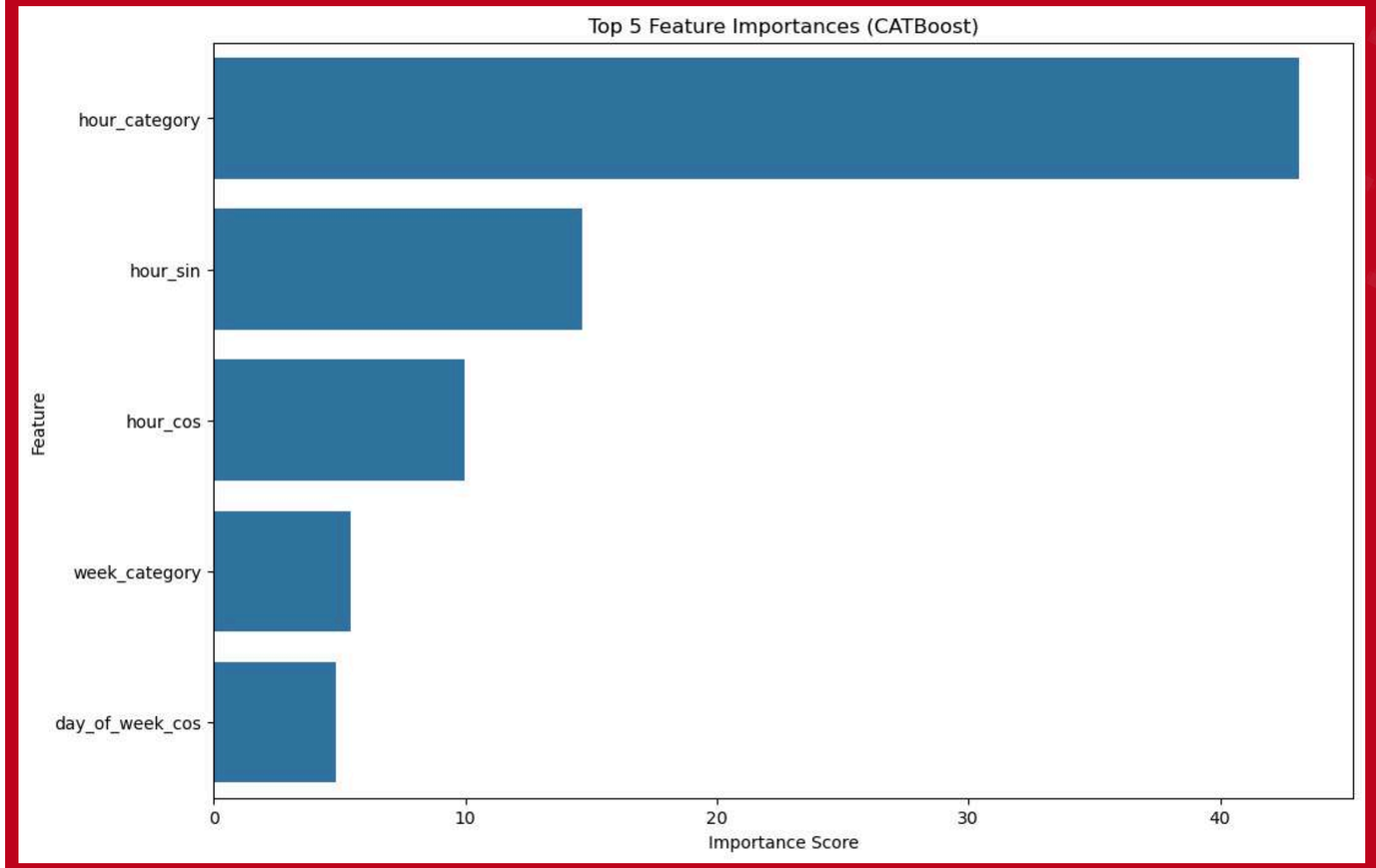
The Best Hyperparameter Tunning

Best Parameter combination: **depth**=10 and **iterations**=500 allow the model to learn complex patterns, **learning_rate**=0.1 maintains learning stability, while **l2_leaf_reg**=5 and **random_strength**=10 help prevent overfitting. The **border_count**=128 value indicates the importance of fine-grained discretization of numeric features, and **bagging_temperature**=1 provides variation in the boosting process. This combination produces a model that is robust, accurate, and able to capture non-linear patterns effectively.

```
CATBoost
Best_score: -0.30141209641540206
Best_params: {'random_strength': 10, 'learning_rate': 0.1, 'l2_leaf_reg': 5, 'iterations': 500, 'depth': 10, 'border_count': 128, 'bagging_temperature': 1}
```

Feature Importances

The feature importance results show that time variables such as '**hour_category**', '**hour_sin**', and '**hour_cos**' have the **greatest influence** on the prediction, indicating that the time of day greatly determines **the amount of rental/usage**. In addition, weekly features such as '**week_category**' and '**day_of_week_cos**' also contribute significantly, indicating the presence of weekly seasonal patterns. Thus, **the time component is the main factor that shapes the target behavior pattern in this data.**

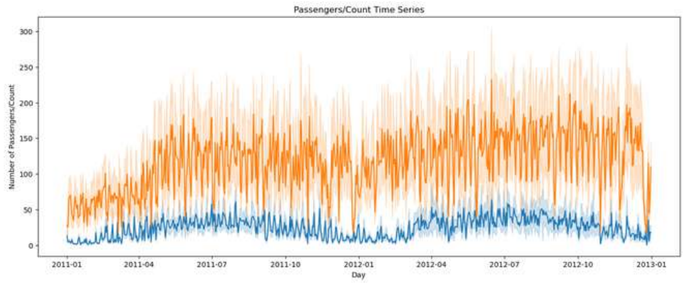


Model Limitation

	Range	# Score MAE	# Score MAPE	# Score RMSE	# Score R2
0	<=50	5.220548934160844	0.4148556951146379	70.38543469619954	0.7424154701802148
1	51-100	17.676206935201172	0.21988590115240583	897.0547223248774	0.17929476458619353
2	101-150	20.902953337975493	0.1720482385487989	860.3573821377602	0.22039807580996051
3	151-200	24.596861519902763	0.14134231160864807	1327.7315968895305	0.15036354397520957
4	201-250	26.635824520208185	0.12422214891420591	1402.081432740753	0.0852973776959135
5	251-300	32.50785618803916	0.11984937126602938	1842.7119230512494	0.11914127437958633
6	301-350	34.78911900987872	0.10414056172048627	2239.271596030173	0.09823640436313297
7	351-400	40.15213325642647	0.10528135659269695	2650.828504654054	-0.036279760080113466
8	401-450	28.87054760290319	0.06804986980363563	1282.6343076712367	0.06136836037166138
9	451-500	20.153889149662188	0.04411295775698867	604.4807322264767	0.10145704651040455
10	>501	51.14943040687052	0.10291635896754633	2616.264230947291	*Missing value*
11	All Count Range (Max 548)	18.76551469365117	0.2351921383722612	929.0567425689343	0.936817474135186

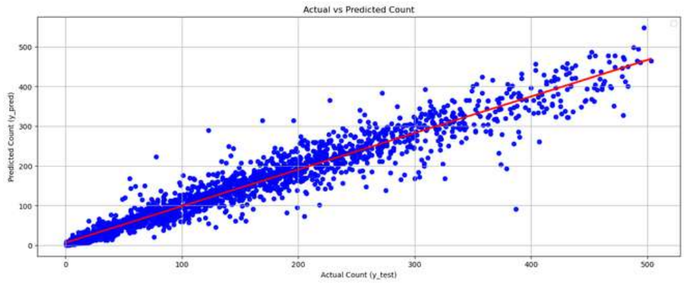
This model is recommended for use on a range of values that are **not too large** (<=350 is better) in order to maintain high prediction accuracy. For larger prediction values, the model can still be used, but it is recommended to **perform a sanity check** to ensure that the **prediction results remain logical and acceptable**.

RECOMMENDATION



Focus on Time Variable

Features such as hour_category, hour_sin, and week_category have a significant impact on predictions. It is recommended that decision making or resource allocation take these time patterns into account.



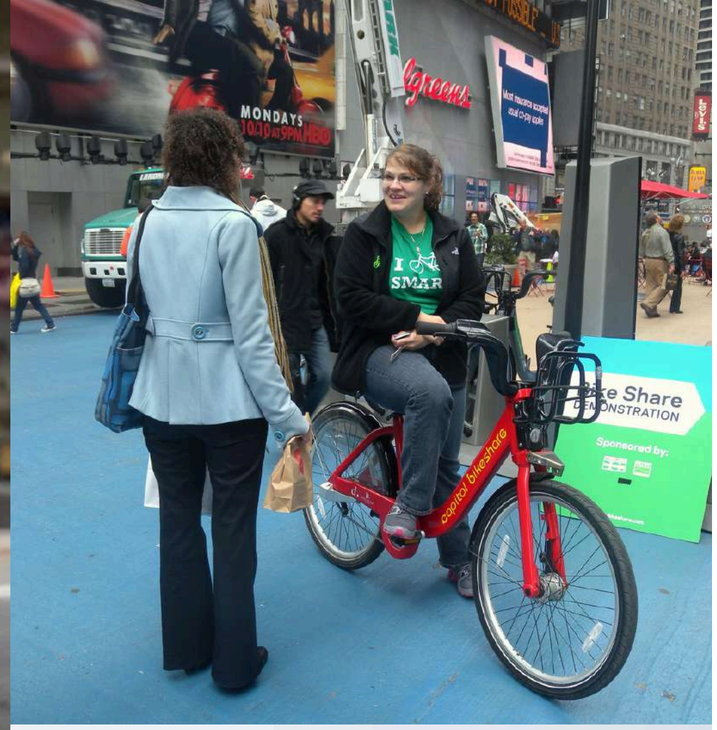
Beware of Inaccuracies in Big Predictions

The model performs lower than the high predicted values (>350), so prediction results in this range should be reviewed or used with caution.

```
--- Regression Model Metrics ---
Mean Absolute Error (MAE)                  : 18.77
Root Mean Squared Error (RMSE)             : 30.48
R-squared (R2 Score)                       : 0.9368
Mean Absolute Percentage Error (MAPE)      : 0.2352
--------------------------------------------------------------------
```

Adoption of CatBoost Model as Production Model

With high accuracy ($R^2 \approx 0.93$) and low error, the CatBoost model is very suitable for application in operational prediction systems.



Cutting Cost: Operational Efficiency

With an MAE of 18.77 and an R^2 of 0.9368, this model is **able** to predict **demand or volume**.

- Companies can allocate resources more precisely (e.g. fleet, staff, inventory).
- Avoid waste due to overestimation of demand.
- Avoid lost opportunities due to underestimation.
- Work shift, maintenance, or supply schedules can be adjusted in a data-driven manner → reducing unnecessary costs.

Increasing Revenue: Service Optimization

- **Adjust dynamic pricing** based on time (time feature is very influential base on Feature Importance) → increase profit margin.
- **Target promotions or discounts** at certain hours/schedules that are predicted to be quiet → maximize capacity.
- **Design products or service packages** that are more in line with customer usage patterns → increase conversion.
- Companies can **reduce idle vehicles** during quiet times and add more during busy times without overstock → save operational costs.

Thank you for listening!