Anthony Kovac - Dorian Dulor - Théo Dutilleul

# Seoul Bike

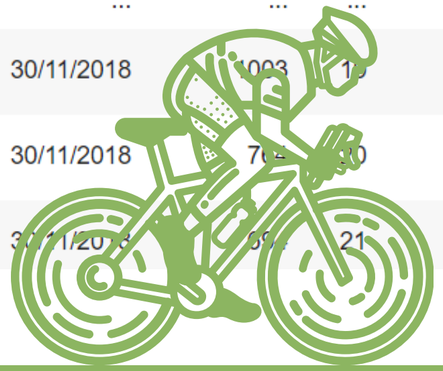
HTTPS://ARCHIVE.ICS.UCI.EDU/ML/DATASETS/SEOUL+BIKE+SHARING+DEMAND

# Summary

- THE DATASET
- THE PROBLEM
- OUR EXPECTATIONS
- OUR RESULTS
- CONCLUSION

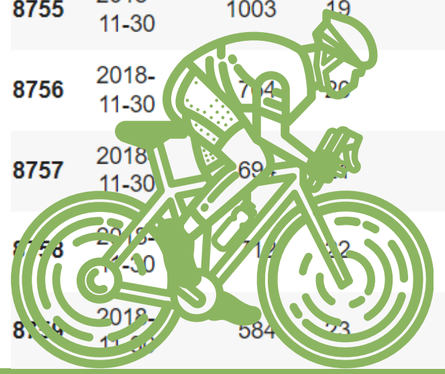
# The Dataset

| | Date | Rented Bike Count | Hour | Temperature(°C) | Humidity(%) | Wind speed (m/s) | Visibility (10m) | Dew point temperature(°C) | Solar Radiation (MJ/m2) | Rainfall(mm) | Snowfall (cm) | Seasons | Holiday | Functioning Day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01/12/2017 | 254 | 0 | -5.2 | 37 | 2.2 | 2000 | -17.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes |
| **1** | 01/12/2017 | 204 | 1 | -5.5 | 38 | 0.8 | 2000 | -17.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes |
| **2** | 01/12/2017 | 173 | 2 | -6.0 | 39 | 1.0 | 2000 | -17.7 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes |
| **3** | 01/12/2017 | 107 | 3 | -6.2 | 40 | 0.9 | 2000 | -17.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes |
| **4** | 01/12/2017 | 78 | 4 | -6.0 | 36 | 2.3 | 2000 | -18.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **8755** | 30/11/2018 | 1003 | 19 | 4.2 | 34 | 2.6 | 1894 | -10.3 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes |
| **8756** | 30/11/2018 | 764 | 20 | 3.4 | 37 | 2.3 | 2000 | -9.9 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes |
| **8757** | 30/11/2018 | 694 | 21 | 2.6 | 39 | 0.3 | 1968 | -9.9 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes |



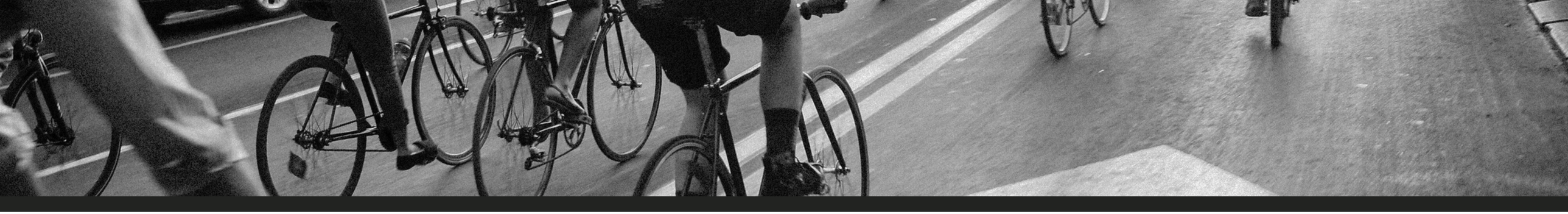
# The Dataset

| | Date | Rented Bike Count | Hour | Temperature(°C) | Humidity(%) | Wind speed (m/s) | Visibility (10m) | Dew point temperature(°C) | Solar Radiation (MJ/m2) | Rainfall(mm) | Snowfall (cm) | Seasons | Holiday | Functioning Day | Day | Month |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2017-12-01 | 254 | 0 | -5.2 | 37 | 2.2 | 2000 | -17.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes | Friday | December |
| **1** | 2017-12-01 | 204 | 1 | -5.5 | 38 | 0.8 | 2000 | -17.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes | Friday | December |
| **2** | 2017-12-01 | 173 | 2 | -6.0 | 39 | 1.0 | 2000 | -17.7 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes | Friday | December |
| **3** | 2017-12-01 | 107 | 3 | -6.2 | 40 | 0.9 | 2000 | -17.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes | Friday | December |
| **4** | 2017-12-01 | 78 | 4 | -6.0 | 36 | 2.3 | 2000 | -18.6 | 0.0 | 0.0 | 0.0 | Winter | No Holiday | Yes | Friday | December |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **8755** | 2018-11-30 | 1003 | 19 | 4.2 | 34 | 2.6 | 1894 | -10.3 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes | Friday | November |
| **8756** | 2018-11-30 | 764 | 20 | 3.4 | 37 | 2.3 | 2000 | -9.9 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes | Friday | November |
| **8757** | 2018-11-30 | 694 | 21 | 2.6 | 39 | 0.3 | 1968 | -9.9 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes | Friday | November |
| **8758** | 2018-11-30 | 712 | 22 | 2.1 | 41 | 1.0 | 1859 | -9.8 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes | Friday | November |
| **8759** | 2018-11-30 | 584 | 23 | 1.9 | 43 | 1.3 | 1909 | -9.3 | 0.0 | 0.0 | 0.0 | Autumn | No Holiday | Yes | Friday | November |



# The Problem

How to accurately predict bike rental demand based on daily parameters and conditions to effectively meet the demand?



## Ins

- Access to historical bike rental data
- Reliable weather data
- Location information
- Time-series data

## Outs

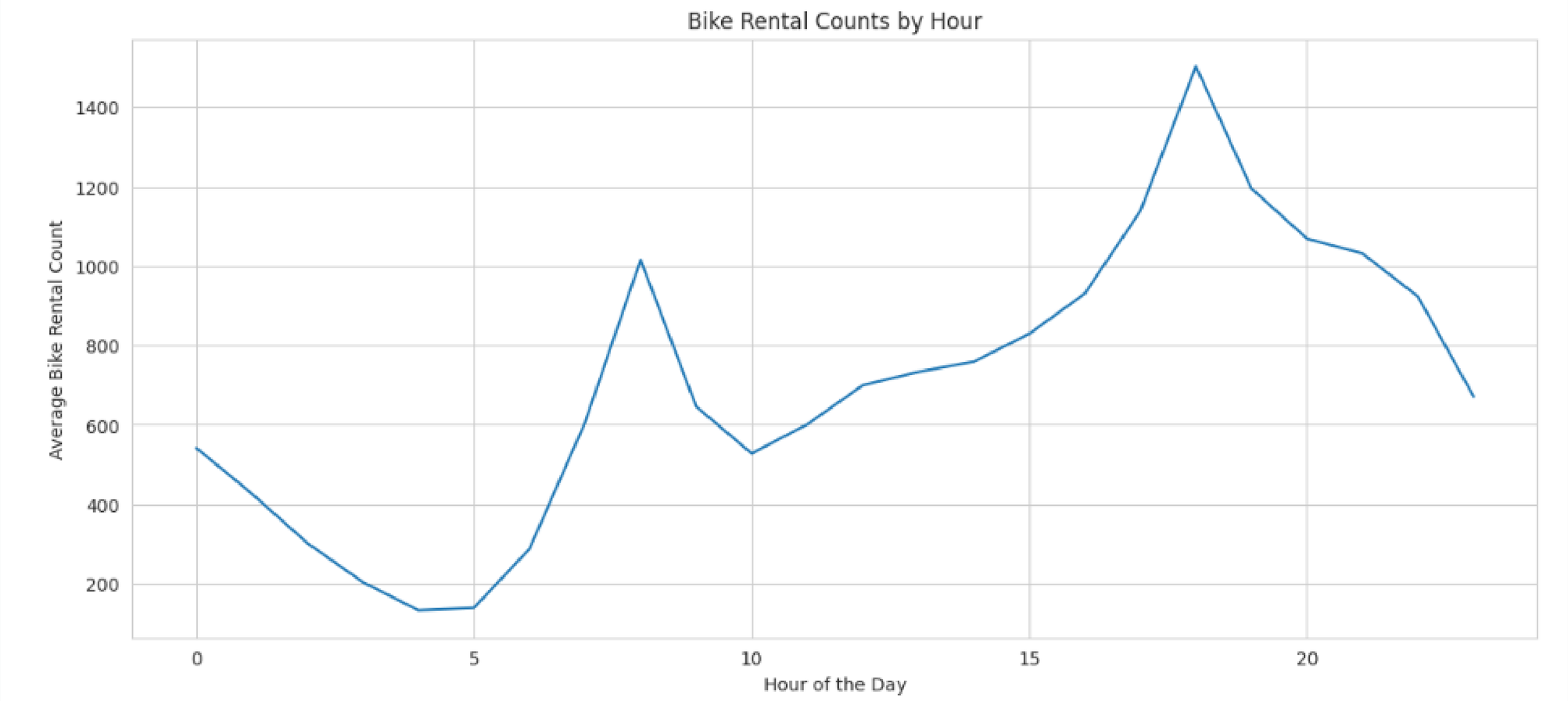
- Data cleaning and processing
- Model selection and evaluation
- Real-time demand forecasting
- Demand-based marketing

# Our Expectations

- Higher demand during weekend and holidays
- Weather and seasons will have a significant impact

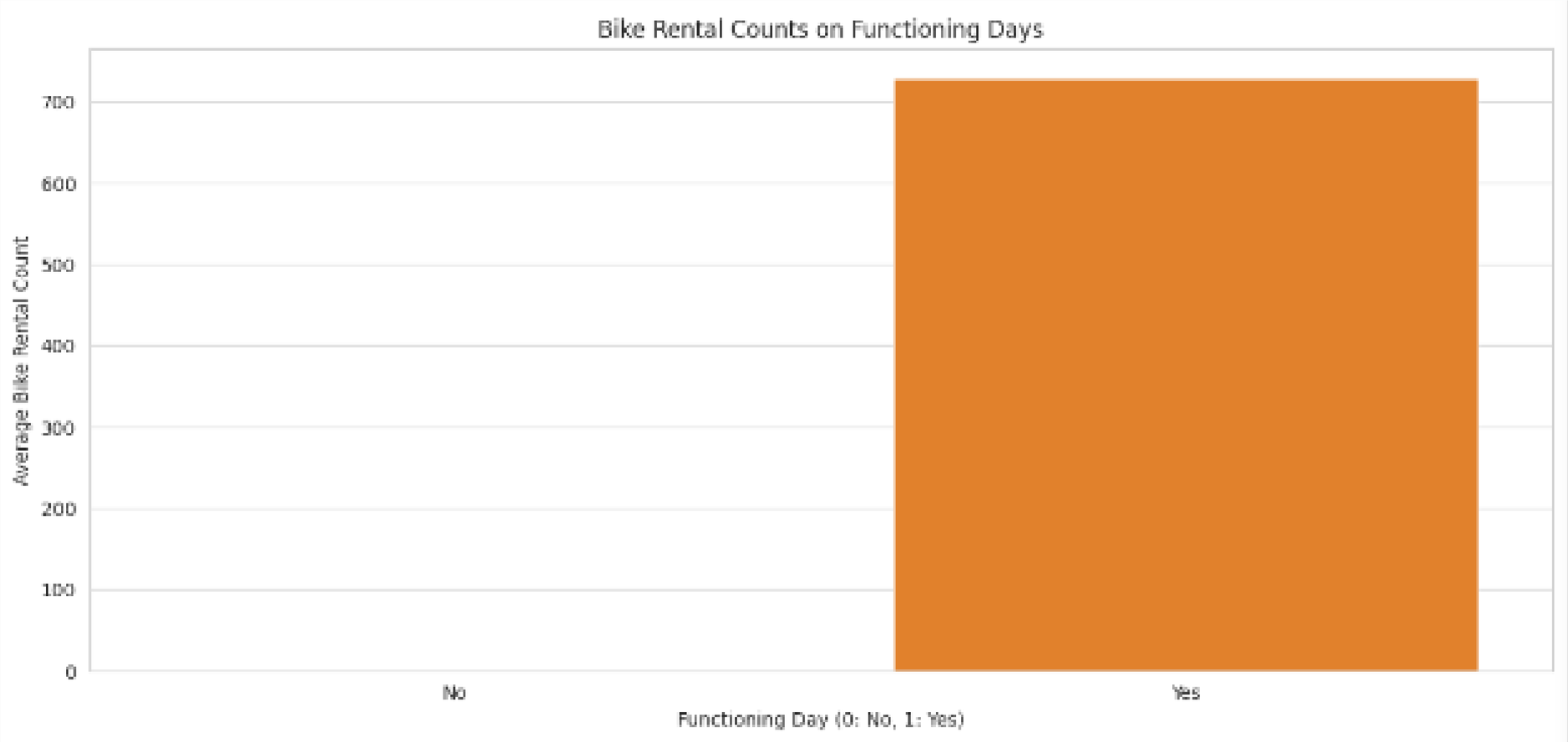
# Our Results

## HOUR OF THE DAY



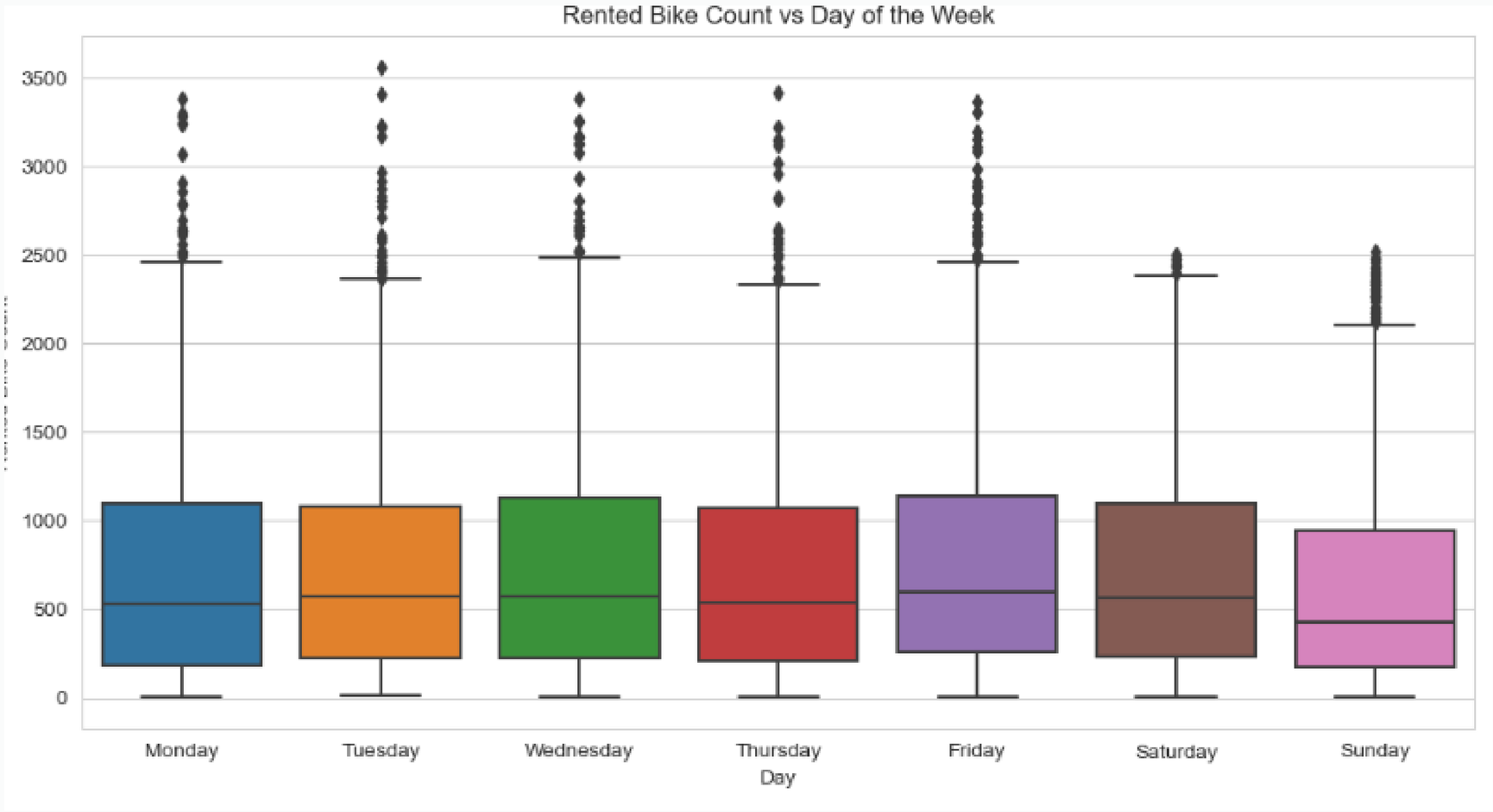
# Our Results

## FUNCTIONING DAYS



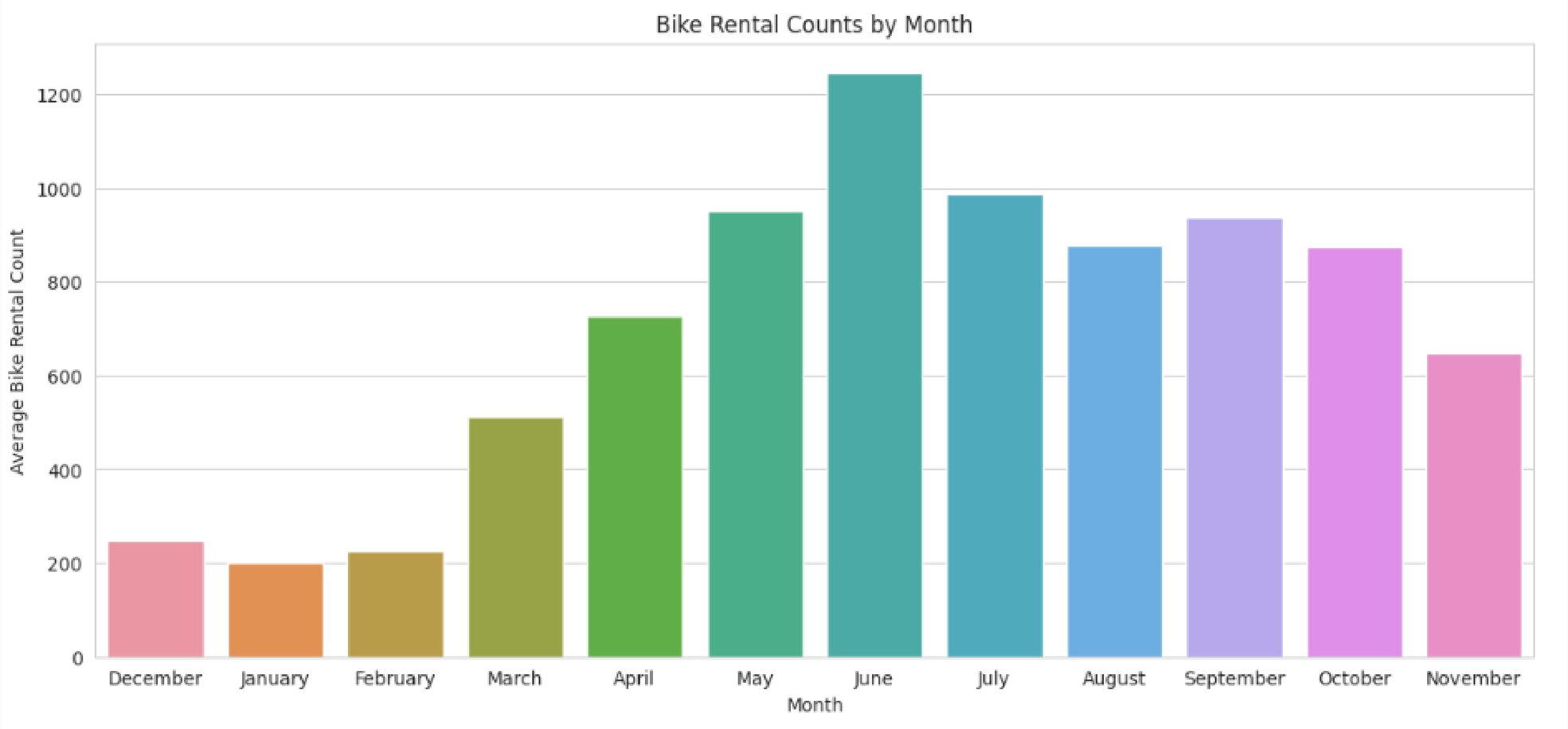
# Our Results

## DAY OF THE WEEK



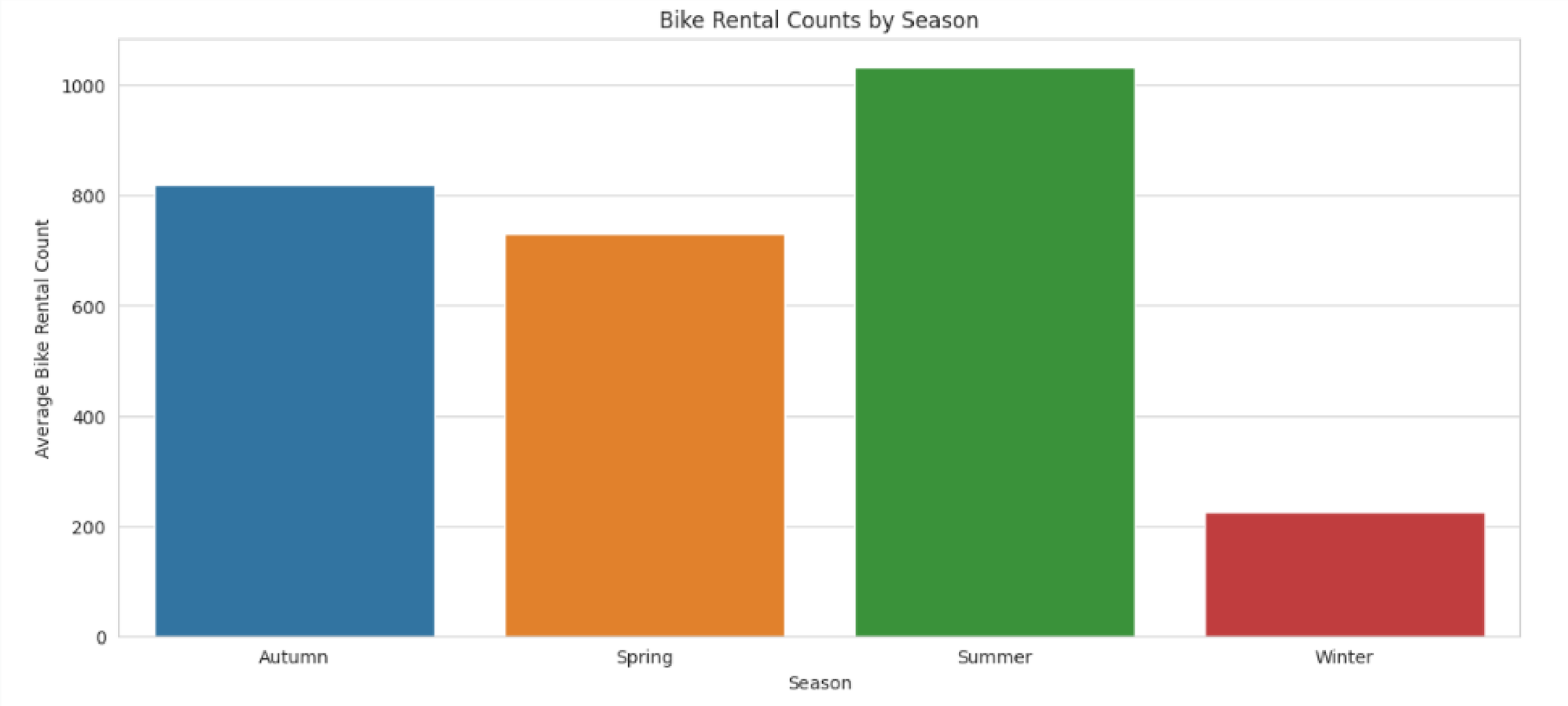
# Our Results

## MONTH



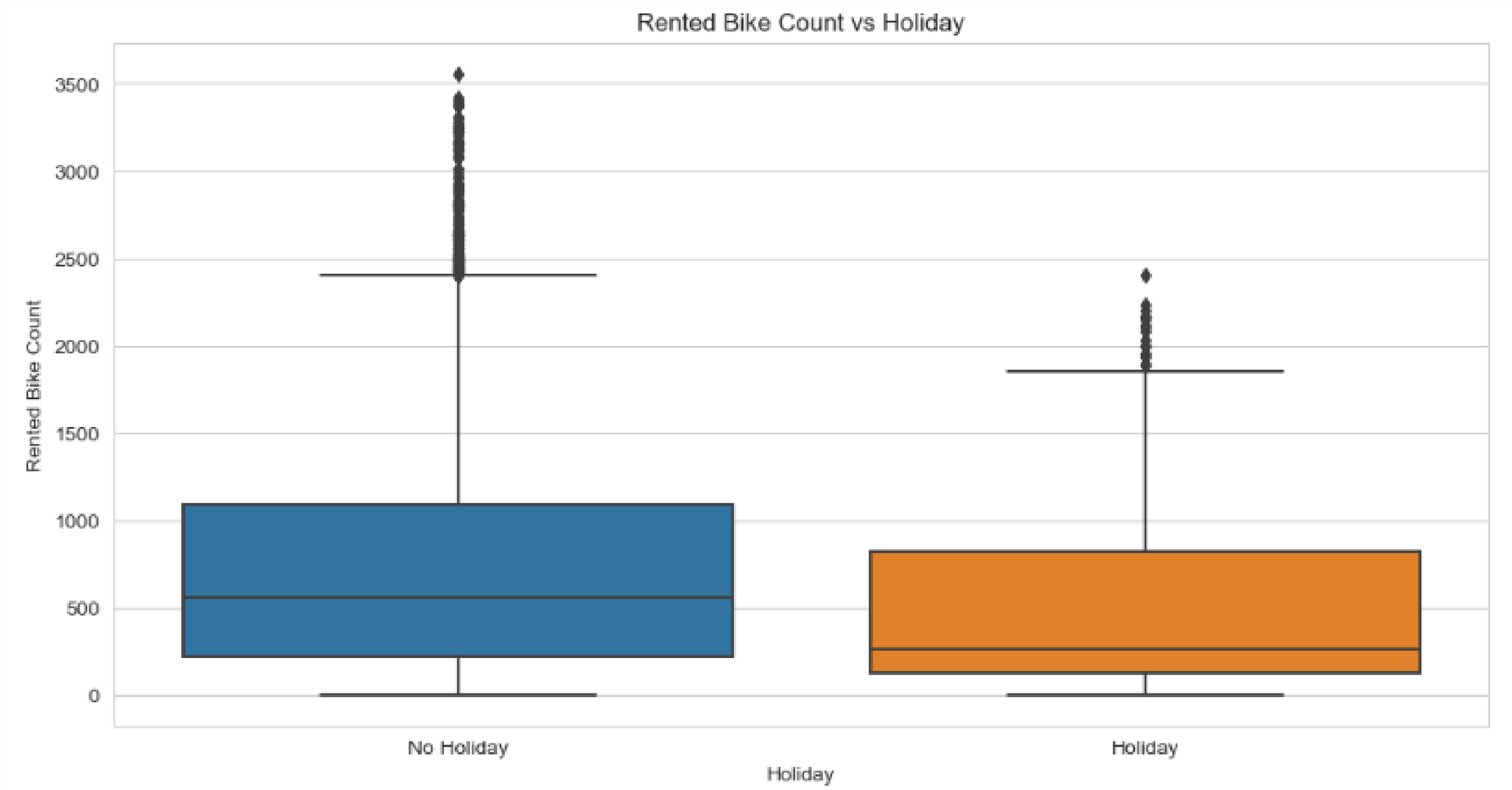
# Our Results

## SEASON



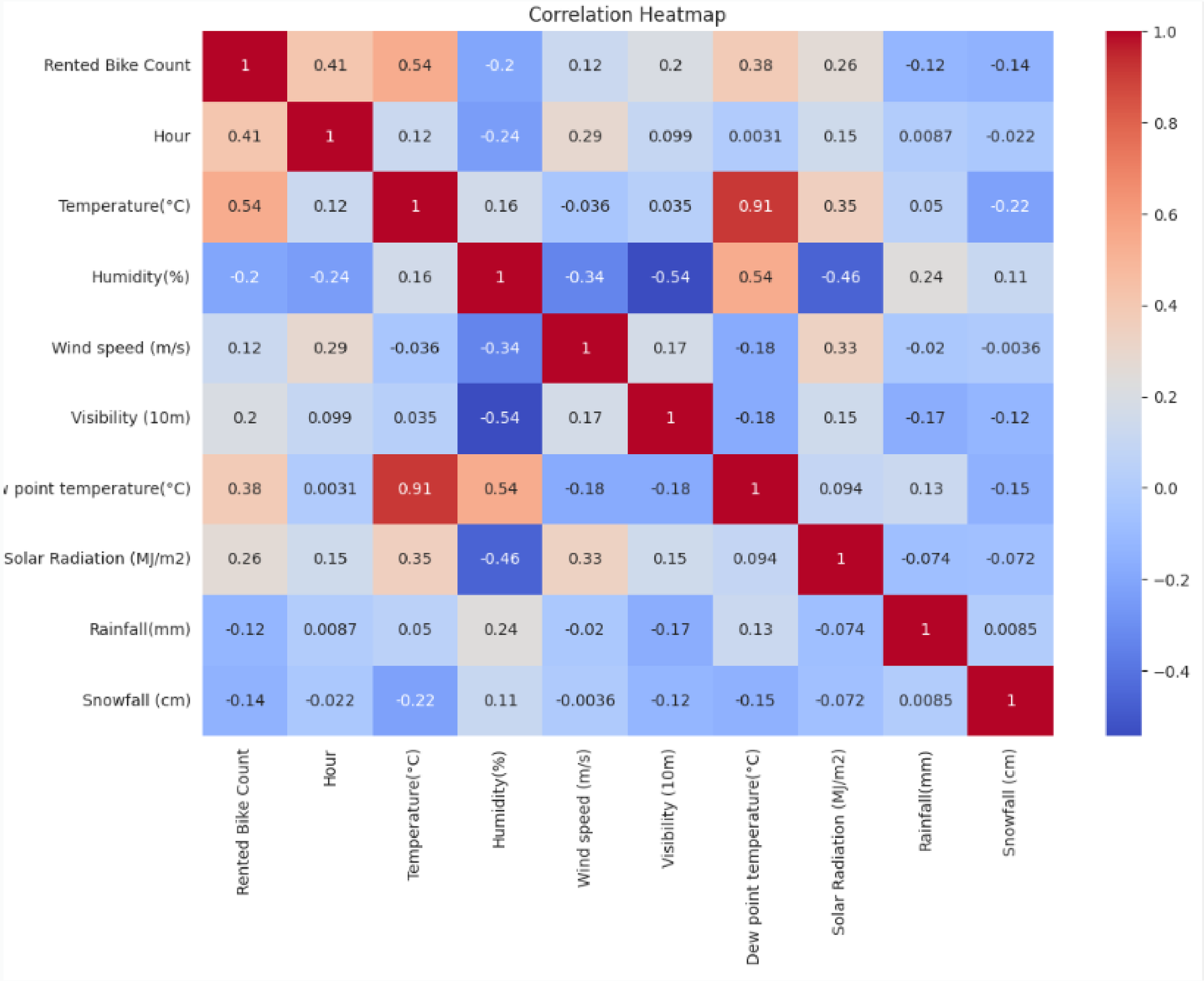
# Our Results

## HOLYDAY



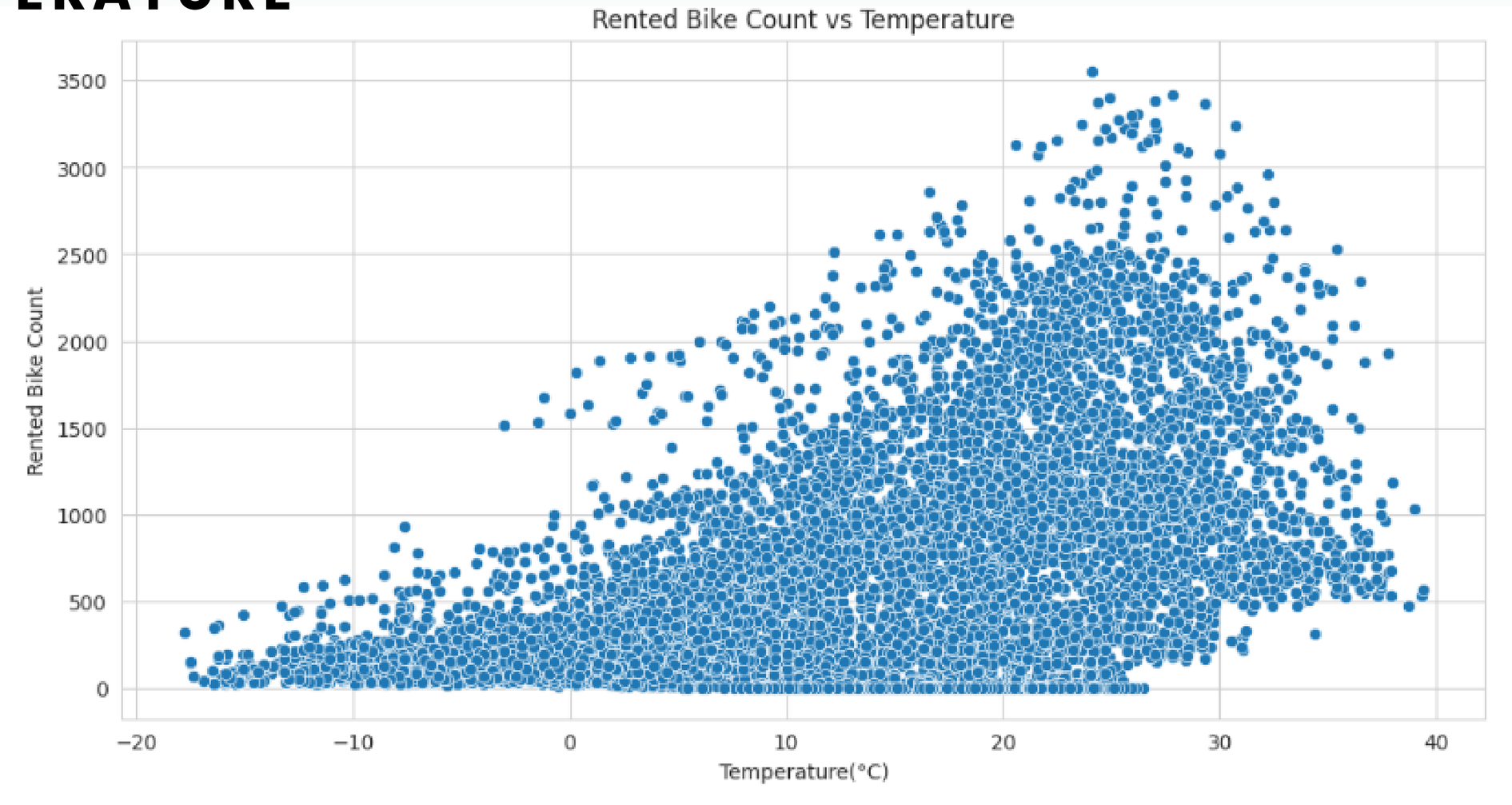
# Our Results

## WEATHER



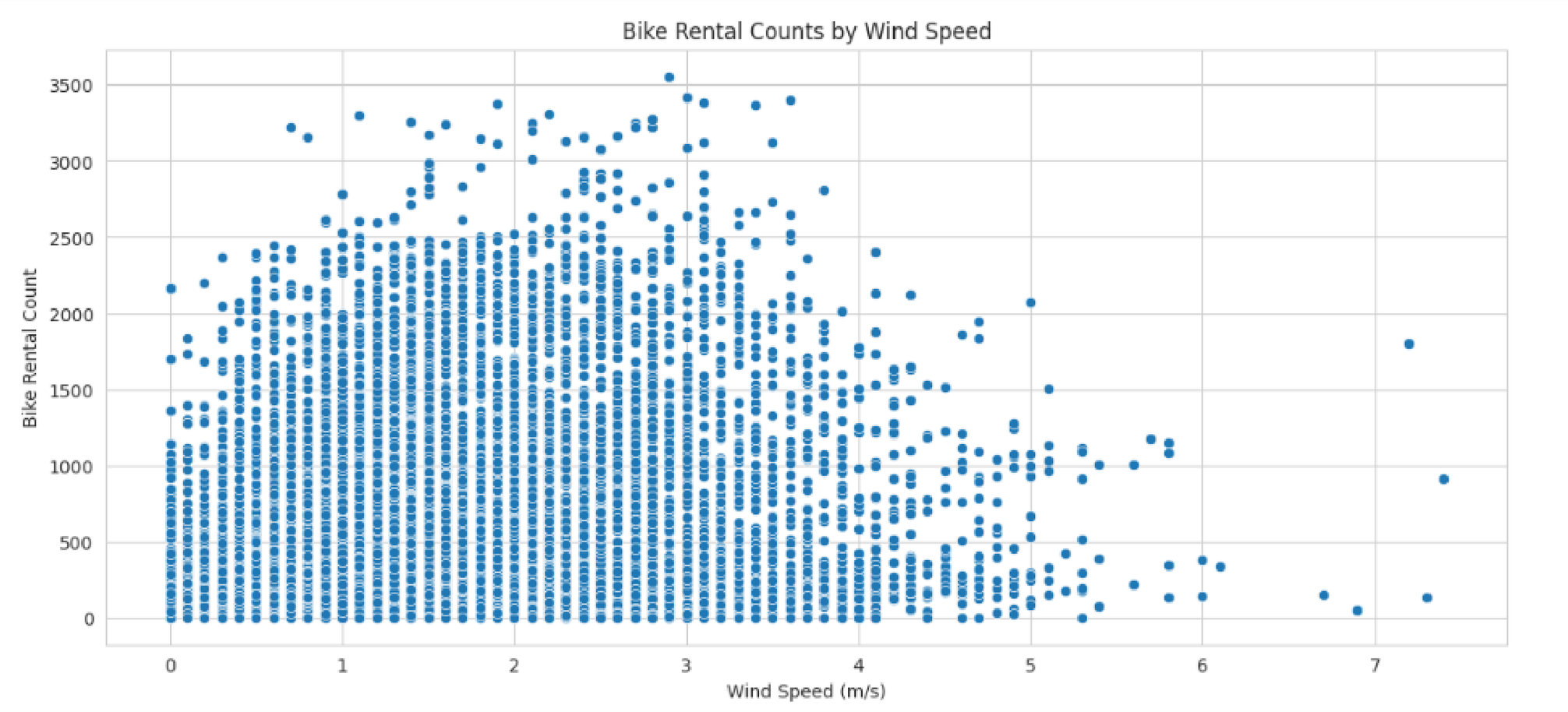
# Our Results

## TEMPERATURE



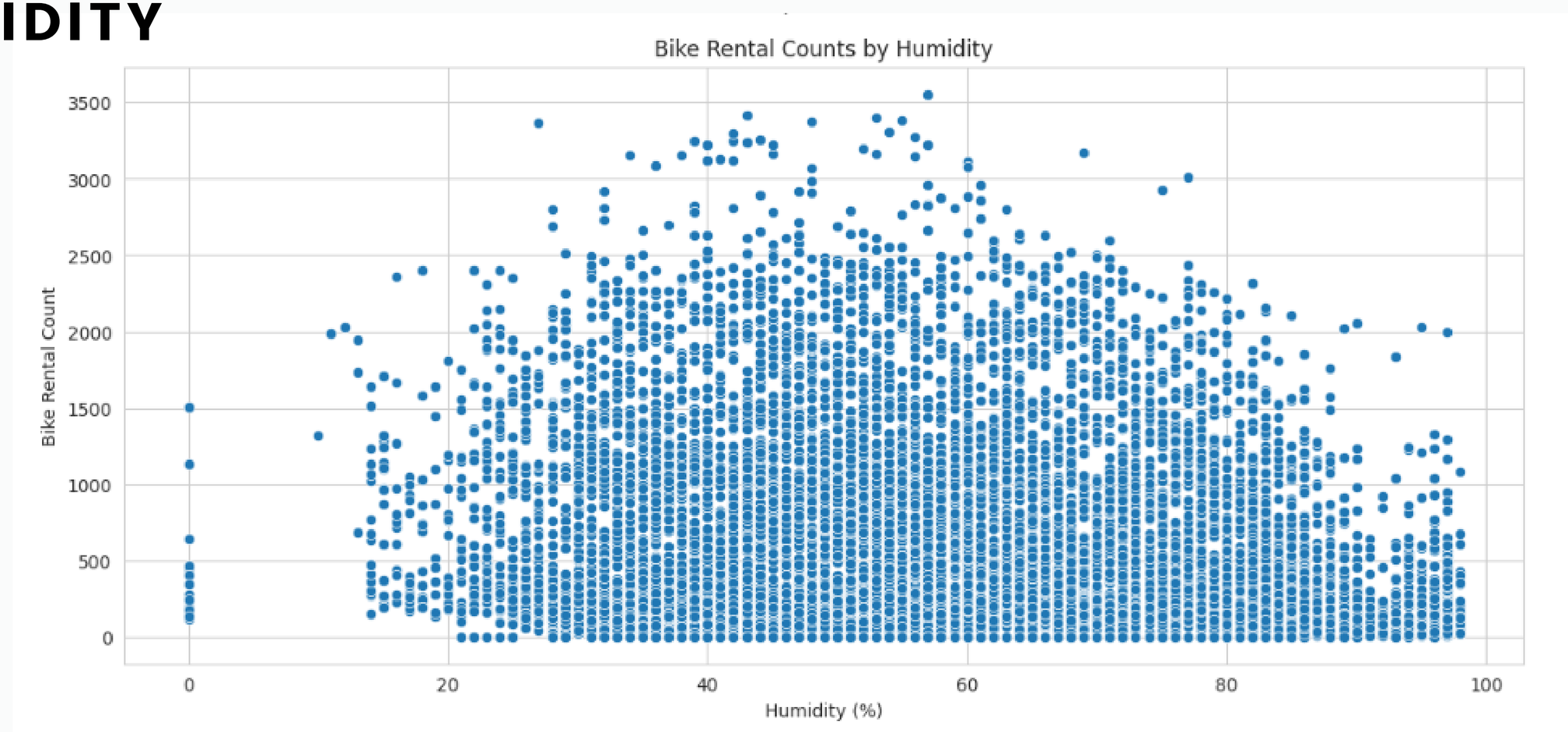
# Our Results

## WIND



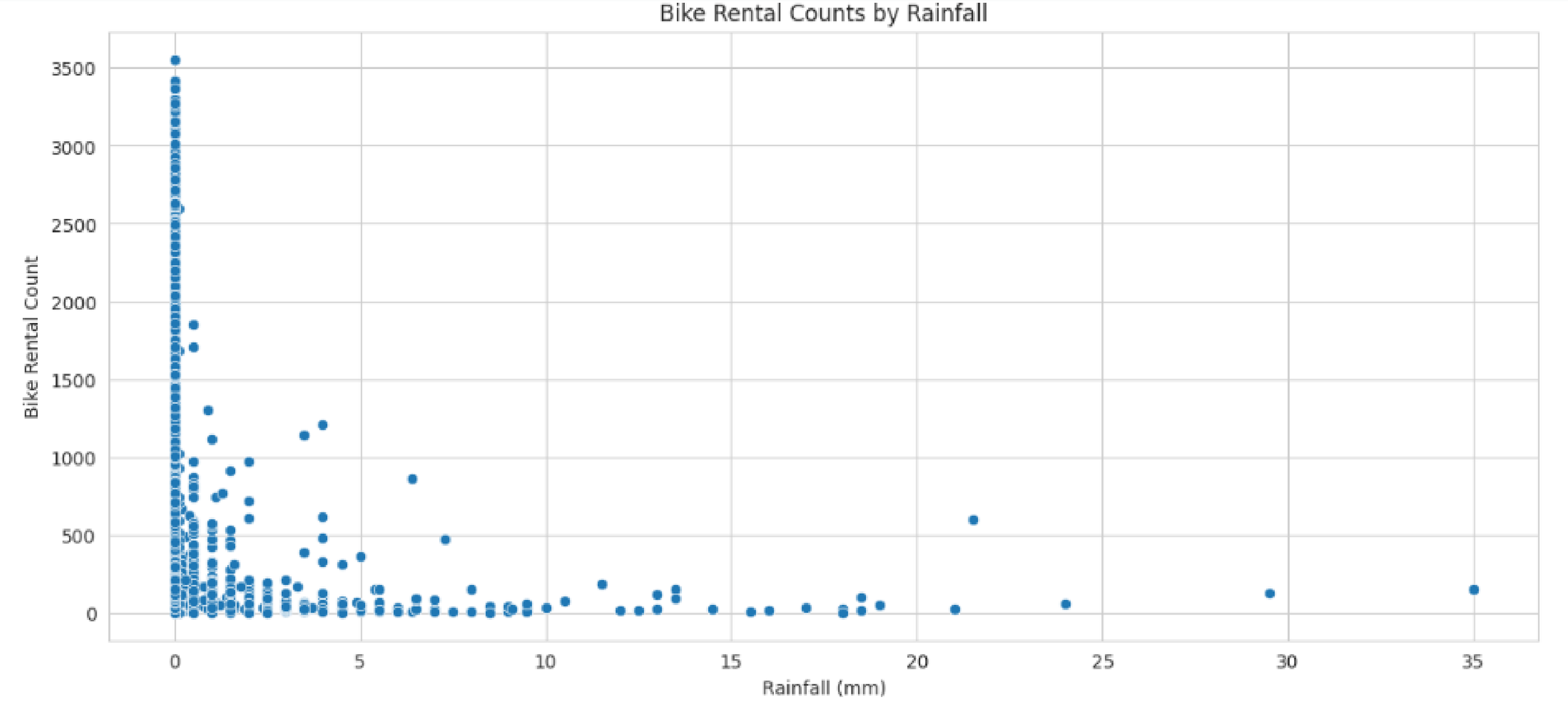
# Our Results

## HUMIDITY



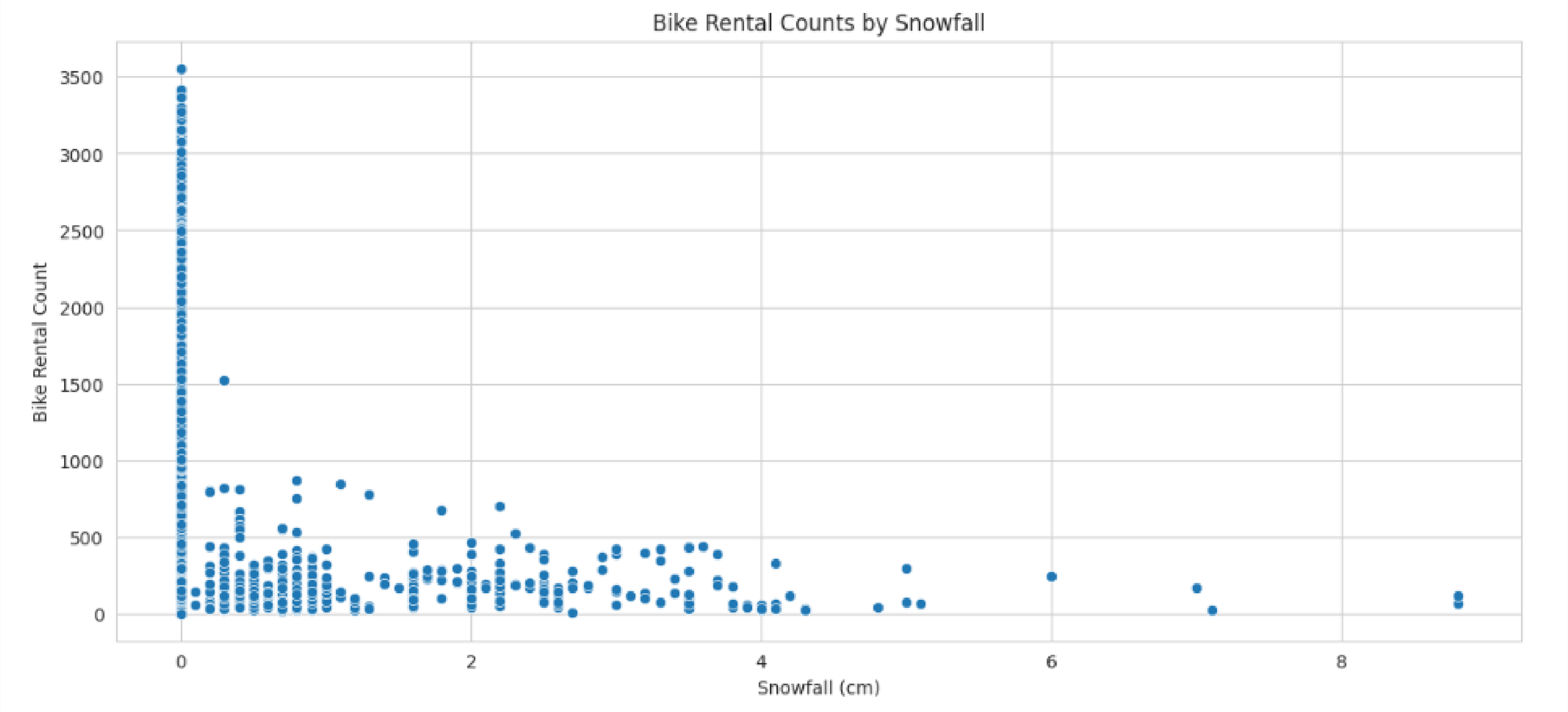
# Our Results

## RAIN



# Our Results

## SNOW



# Our Model

## MODELS WE HAVE TRIED

-Ridge Regression
-Lasso Regression
-Decision Tree Classifier
-Random Forest Classifier
-K-Nearest Neighbors Classifier
-Random Forest Regressor
-Gradient Boosting Regressor
-Linear Regression

# Our Model

## THE MOST EFFICIENT ONE

```
# Find the best performing model based on R2 score
best_model = max(models_performance, key=lambda x: models_performance[x]['R2'])

models_performance
```

```
# Random Forest Regressor
rf = RandomForestRegressor()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
models_performance['Random Forest Regressor'] = {'MSE': mse_rf, 'R2': r2_rf}
```

# Our Model

## HYPERPARAMETERS OPTIMISATION

```
[ ] # from sklearn.model_selection import GridSearchCV

    # # Define a grid of hyperparameters to search
    # param_grid = {
    #     'n_estimators': [100, 200, 300],
    #     'max_depth': [10, 20, None],
    #     'min_samples_split': [2, 5, 10],
    #     'min_samples_leaf': [1, 2, 4],
    #     'max_features': ['auto', 'sqrt']
    # }

    # rf = RandomForestRegressor()

    # # Grid Search model
    # grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2, scoring='r2')
    # grid_search.fit(X_train, y_train)

    # # Get the best estimator
    # best_rf = grid_search.best_estimator_

    # # Prediction
    # y_pred_best_rf = best_rf.predict(X_test)

    # mse_best_rf = mean_squared_error(y_test, y_pred_best_rf)
    # r2_best_rf = r2_score(y_test, y_pred_best_rf)

    # best_rf_params = grid_search.best_params_
    # best_rf_score = grid_search.best_score_

    # best_rf_params, best_rf_score, mse_best_rf, r2_best_rf
```

```
# Manually tuned Random Forest Regressor
rf_manual = RandomForestRegressor(
    n_estimators=150,
    max_depth=None,
    min_samples_split=5,
    min_samples_leaf=2,
    max_features='auto',
)

# Fit the manually tuned model to the training data
rf_manual.fit(X_train, y_train)

# Predict on the test set with the manually tuned model
y_pred_rf_manual = rf_manual.predict(X_test)

# Calculate the performance metrics for the manually tuned model
mse_rf_manual = mean_squared_error(y_test, y_pred_rf_manual)
r2_rf_manual = r2_score(y_test, y_pred_rf_manual)

# Output the performance metrics
mse_rf_manual, r2_rf_manual
```

# Our Model

## HOW IT WORKS

```python
input_data = {
    'Hour': [13],  # Heure de la journée (13h)
    'Temperature(°C)': [22.3],  # Température en Celsius
    'Humidity(%)': [56],  # Pourcentage d'humidité
    'Wind speed (m/s)': [2.3],  # Vitesse du vent (m/s)
    'Visibility (10m)': [2000],  # Visibilité (10m)
    'Dew point temperature(°C)': [14.2],  # Température du point de rosée
    'Solar Radiation (MJ/m2)': [0.5],  # Radiation solaire (MJ/m2)
    'Rainfall(mm)': [0],  # Précipitations (mm)
    'Snowfall (cm)': [0],  # Chutes de neige (cm)
    'Seasons': ['Autumn'],  # Saisons
    'Holiday': ['Holyday'],  # Vacances
    'Functioning Day': ['Yes'],  # Jour de fonctionnement
    'Date': ['01/02/2023']  # Date
}
```

```python
# Transformer 'Date' en 'Day' et 'Month'
input_df = pd.DataFrame(input_data)
input_df['Date'] = pd.to_datetime(input_df['Date'], format='%d/%m/%Y')
input_df['Day'] = input_df['Date'].dt.day_name()
input_df['Month'] = input_df['Date'].dt.month_name()

# Clé pour appliquer la prédiction seulement si le shop est ouvert
key = input_df['Functioning Day'][0] == 'Yes'

input_df.drop(['Date', 'Functioning Day'], axis=1, inplace=True)

# Encodage des variables catégorielles
input_df_encoded = pd.get_dummies(input_df, drop_first=False)

# Ajouter manuellement les colonnes manquantes et les initialiser à 0
expected_features = [
    'Hour', 'Temperature(°C)', 'Humidity(%)', 'Wind speed (m/s)',
    'Visibility (10m)', 'Dew point temperature(°C)', 'Solar Radiation (MJ/m2)',
    'Rainfall(mm)', 'Snowfall (cm)', 'Seasons_Autumn', 'Seasons_Spring',
    'Seasons_Summer', 'Seasons_Winter', 'Holiday_Holiday', 'Holiday_No Holiday', 'Day_Friday', 'Day_Monday',
    'Day_Saturday', 'Day_Sunday', 'Day_Thursday', 'Day_Tuesday', 'Day_Wednesday',
    'Month_April', 'Month_August', 'Month_December', 'Month_February',
    'Month_January', 'Month_July', 'Month_June', 'Month_March', 'Month_May',
    'Month_November', 'Month_October', 'Month_September'
]

for feature in expected_features:
    if feature not in input_df_encoded.columns:
        input_df_encoded[feature] = 0

# Assurer l'ordre correct des colonnes
input_df_encoded = input_df_encoded[expected_features]

# Prédiction
if key:
    prediction = rf_model.predict(input_df_encoded)
    print(prediction)
else:
    print("The shop is closed")
```

# Flask

Hour

Temperature (°C)

Humidity (%)

Wind speed (m/s)

Visibility (10m)

Dew point temperature (°C)

Solar Radiation (MJ/m2)

Rainfall (mm)

Snowfall (cm)

Seasons

Holyday

Functioning Day

Date

Submit

127.0.0.1:5000/predict

Google Workplace  Microsoft Office  ESILV  Dorset


```
{
    "prediction": [
        1187.9859232804233
    ]
}
```

# THANK YOU FOR LISTENING !