Python For Data Analysis

Axel THEVENOT

Seoul Bike Sharing System

Dataset

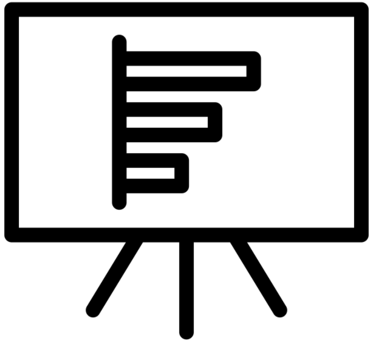
Contains count of public bikes rented **at each hour**

Gives the corresponding informations of

- Weather
- Date
- Holiday

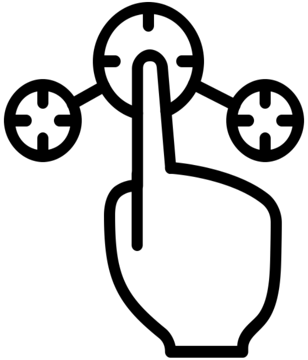
Three Main Steps

Data Analysis



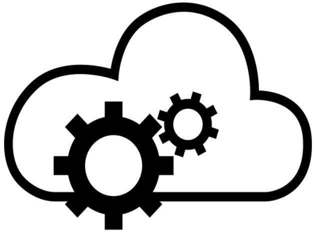
Inspecting, cleaning, transforming, and **modeling** data with the goal of discovering useful **information**, informing conclusions, and supporting decision-making.

Design **experiments** to select a statistical model from a set of candidate models.



Model Selection

Model Deployment



Integrate the selected model into an existing **production** and stable environment where a **client** request the model

Objectives

Predict the number of bikes rented in Seoul based on :

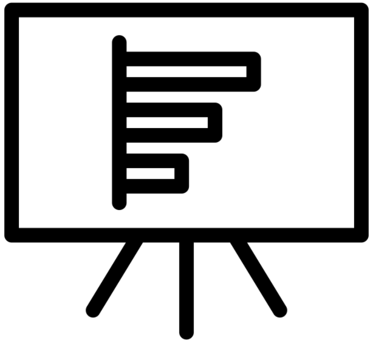
Temporal Features

Date
Hour
Seasons
Holiday

Weather Features

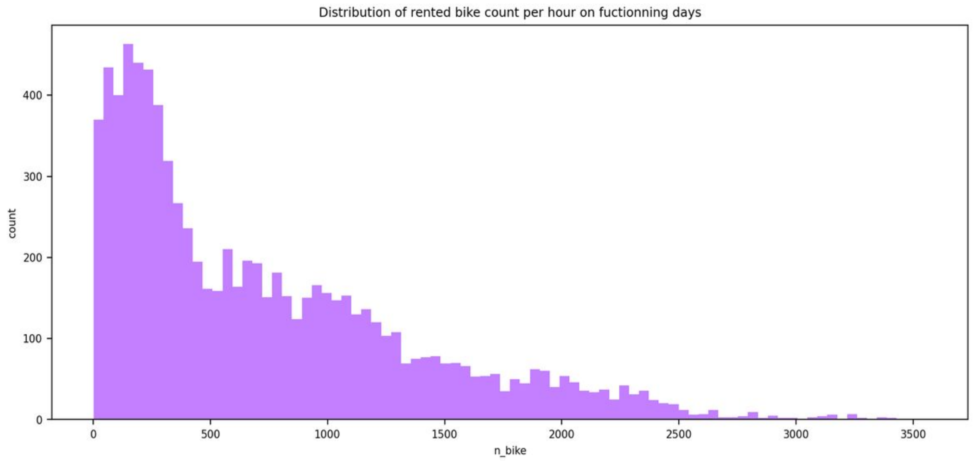
Temperature (°C)
Humidity (%)
Wind speed (m/s)
Visibility (10m)
Dew point temperature (°C)
Solar Radiation (MJ/m2)
Rainfall (mm)
Snowfall (cm)

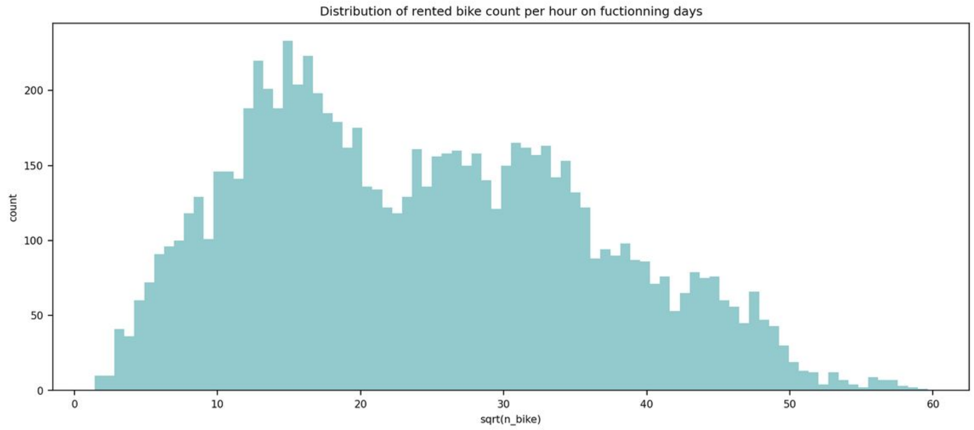
Data Analysis



Inspecting, cleaning, transforming, and **modeling** data with the goal of discovering useful **information**, informing conclusions, and supporting decision-making.

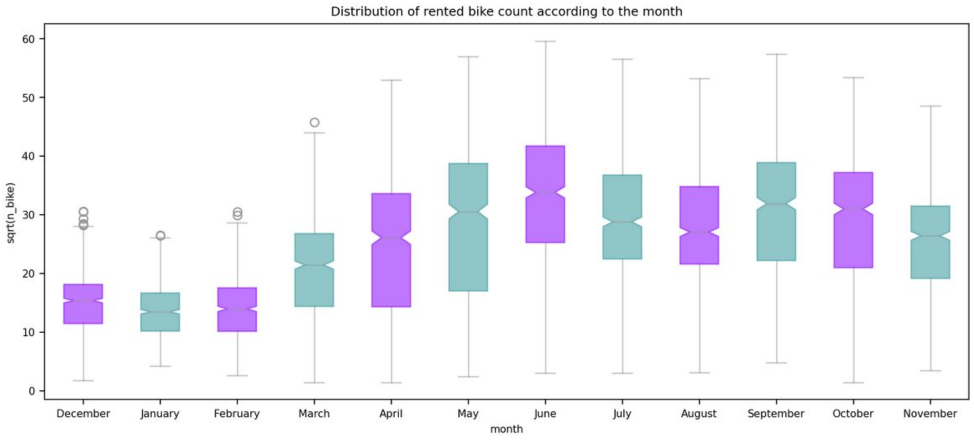
Target : Count Rented Bike

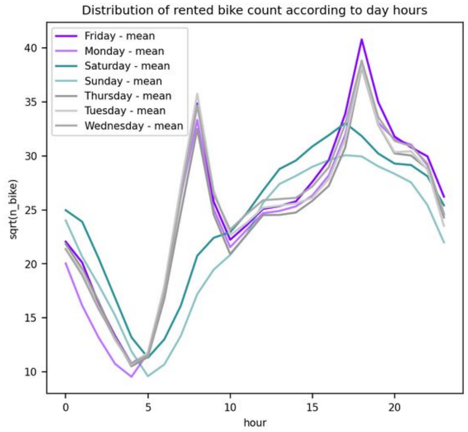


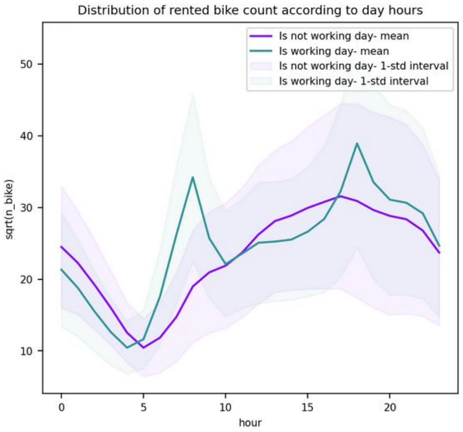


Square root to have **Gaussian-like** distribution and limits the **outliers**

Temporal Features : Date and Hour



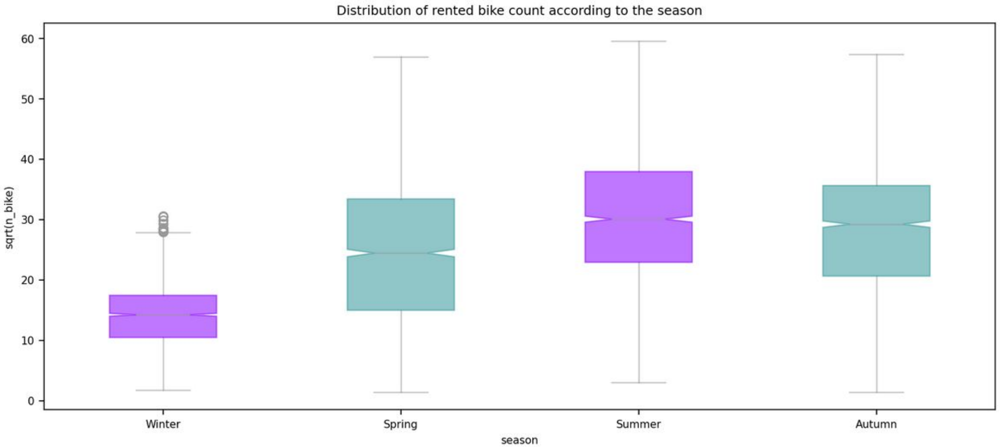


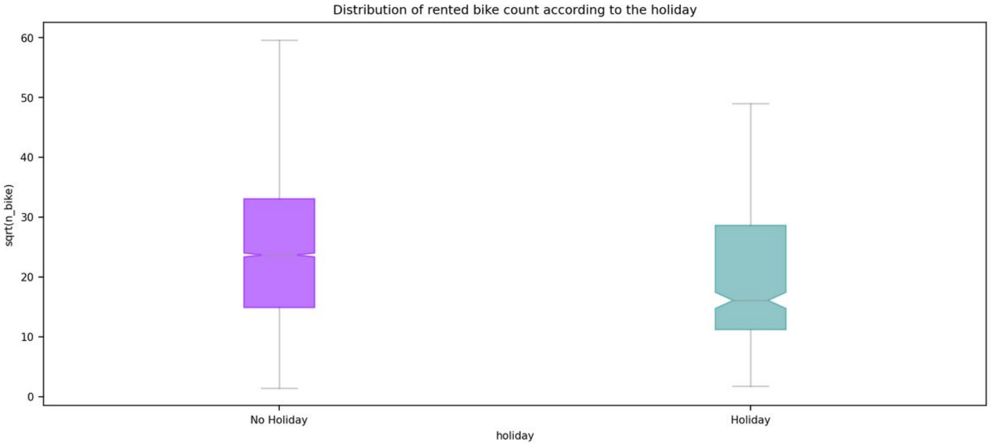


Time serie date and hour changed to **categorical** variable as the dataset contains only one year and the categorical features gives more sense.

Year
Month name
Week day name
Working day condition
Hour

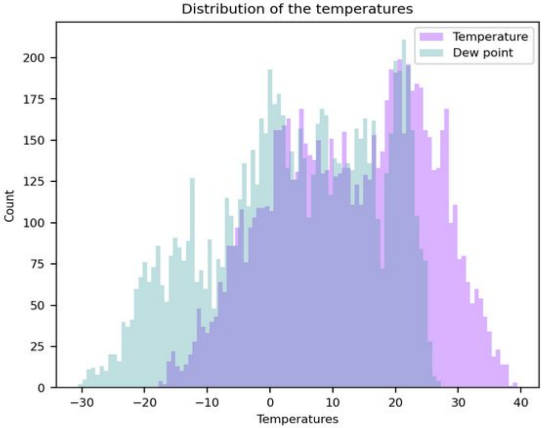
Temporal Features : Seasons and Holiday

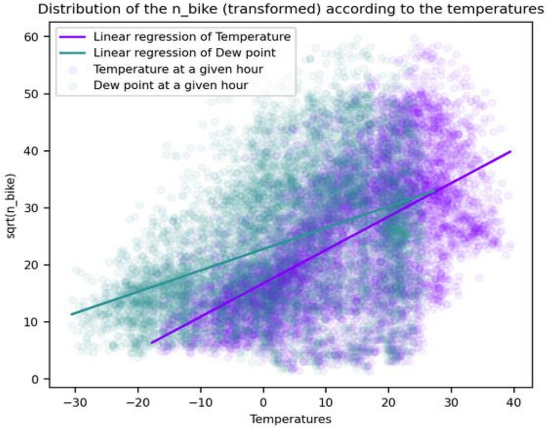


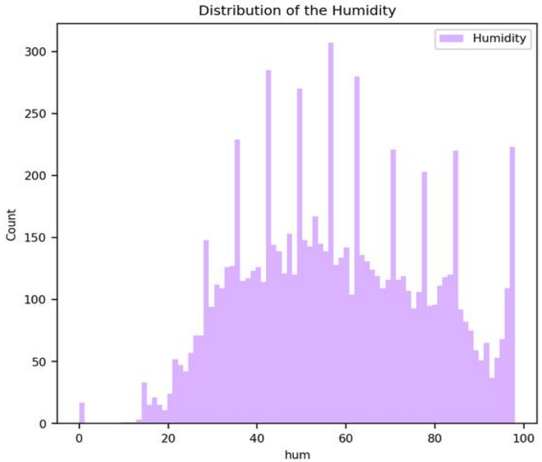


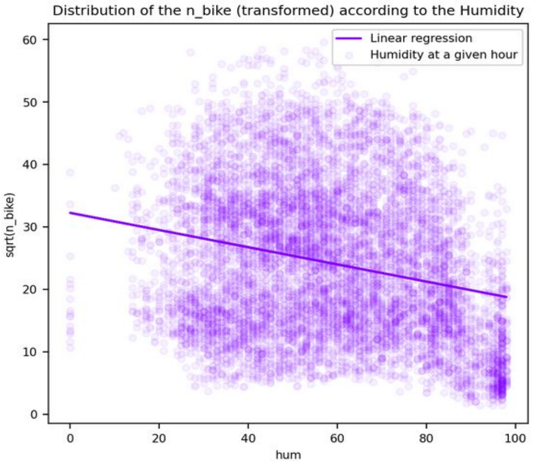
Unchanged

Weather Features : Temperature, Dew, Humidity



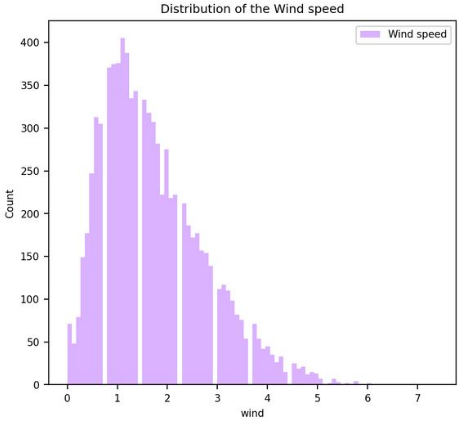


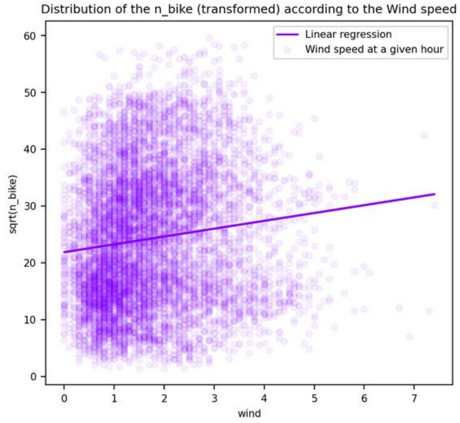




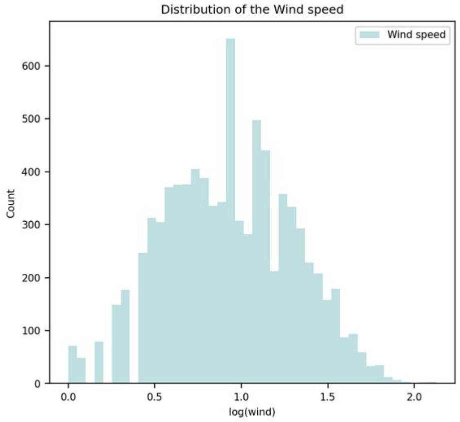
Unchanged

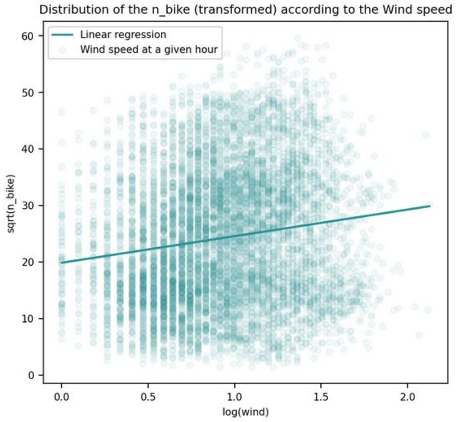
Weather Features : Wind speed



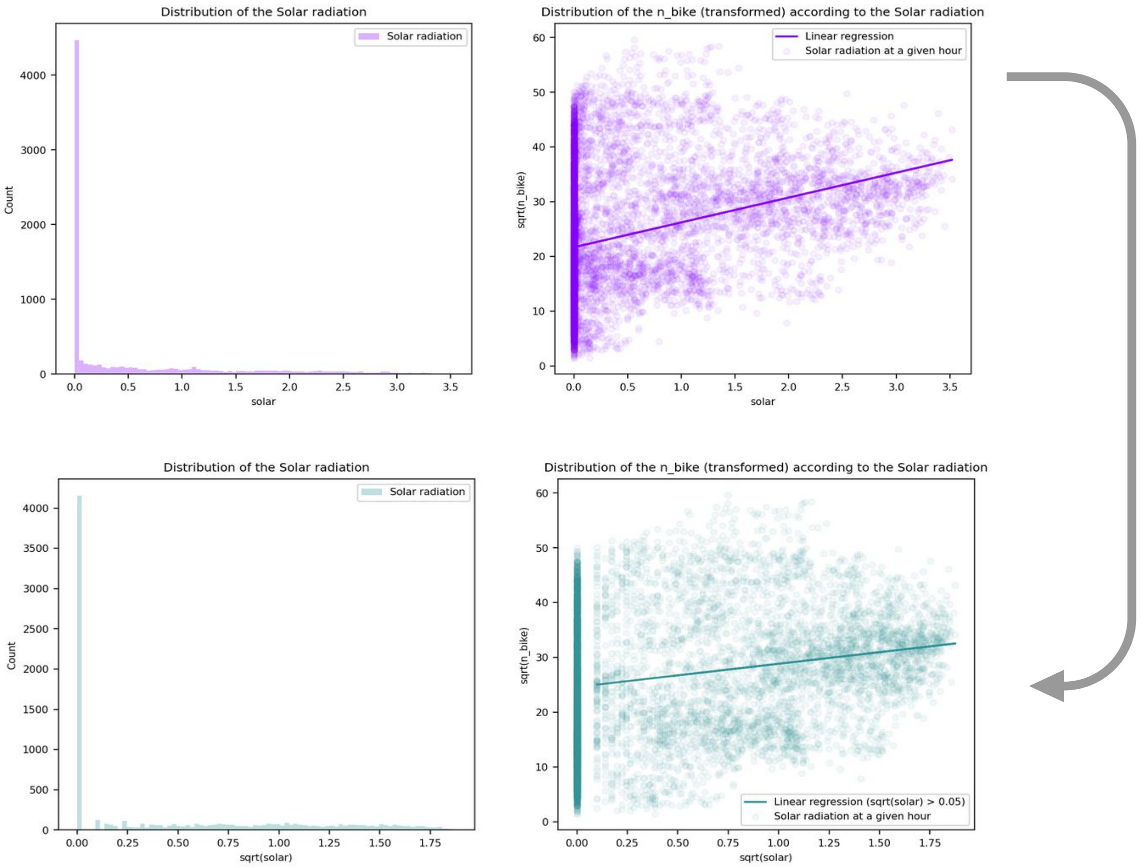


Log to have **Gaussian-like** distribution



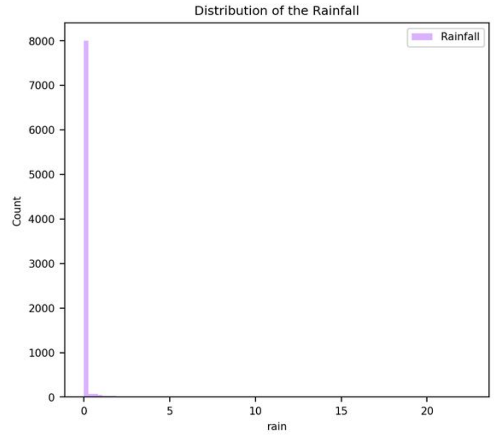


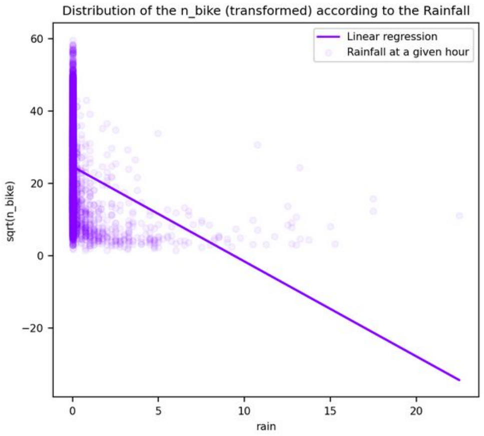
Weather Features : Solar Radiation

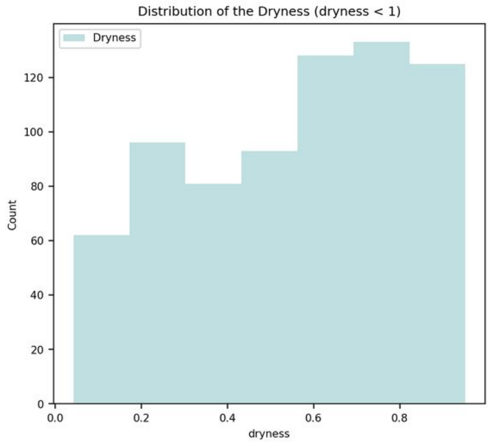


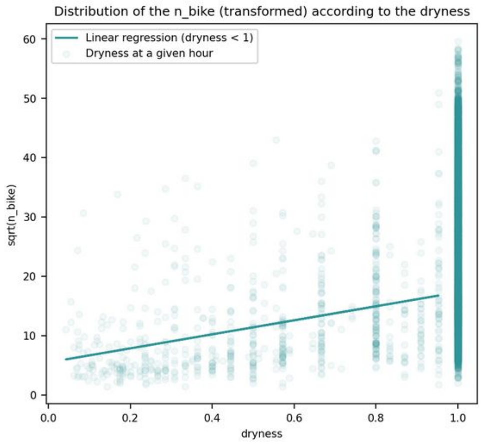
Square root to **Separate** zeros to non zeros

Weather Features : Rainfall



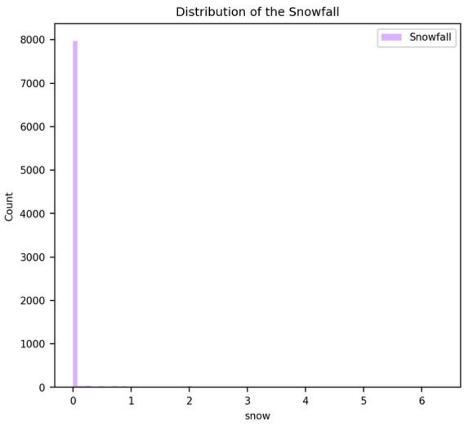


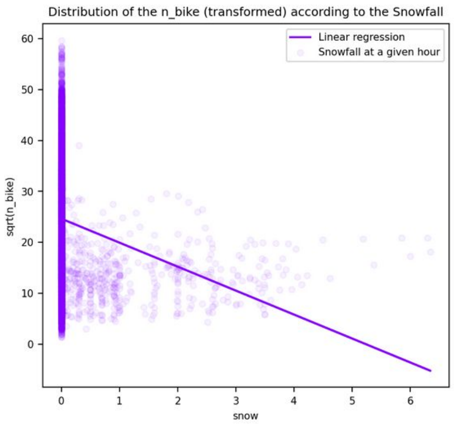


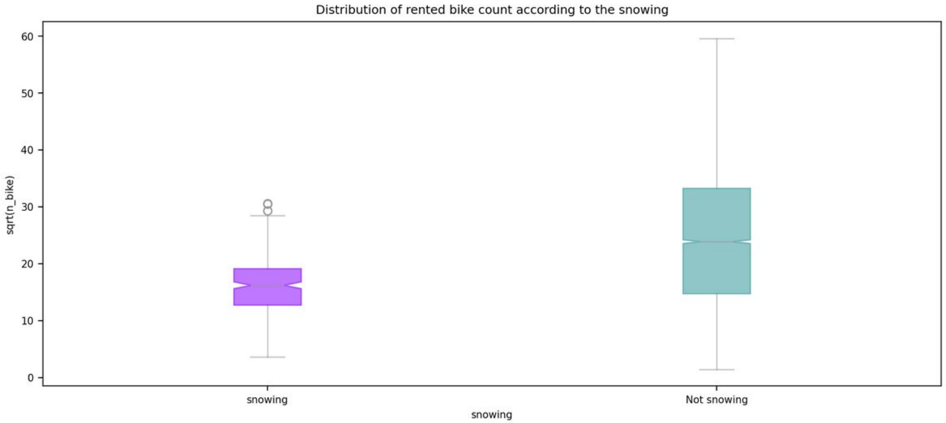


Transformed to **Dryness** of the ground on the two last hours to have a better approximation of biker vision and have a good distribution.

Weather Features : Snowfall

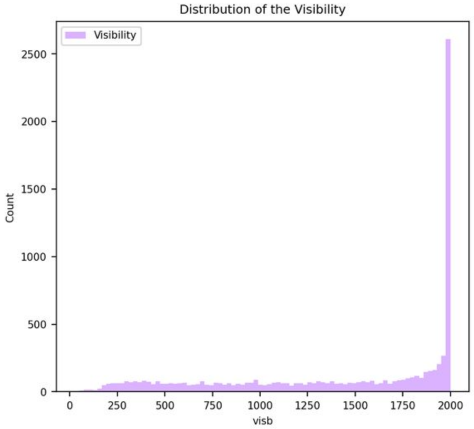


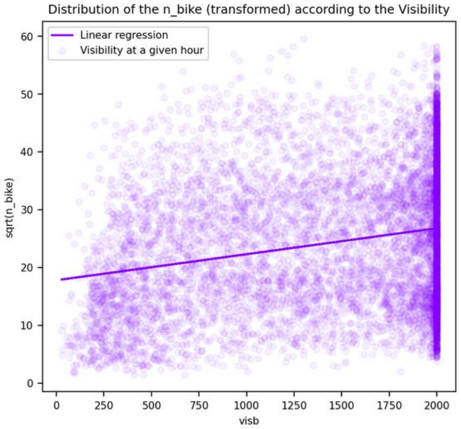


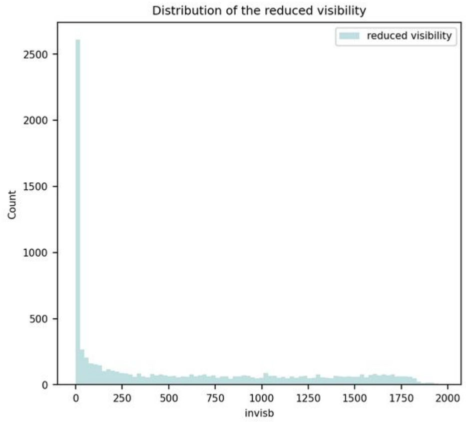


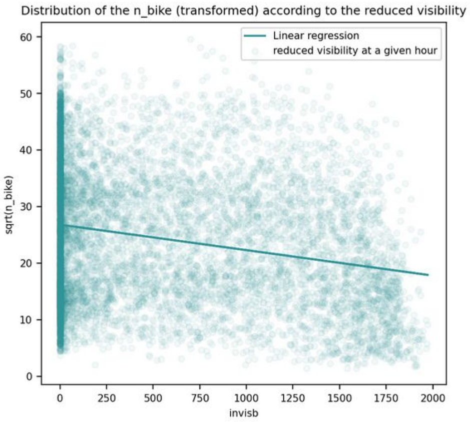
Transformed to **snowing condition** on the 8 last hours to have a better approximation of biker vision.

Weather Features : Visibility









Transformed to **loosed visibility** to have a better approximation of biker vision and put the zero to the perfect condition.

Last step

One hot encode

hour
season
month
day
week_day

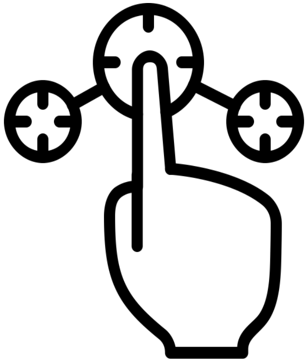
Normalize

humidity
solar radiation
dryness

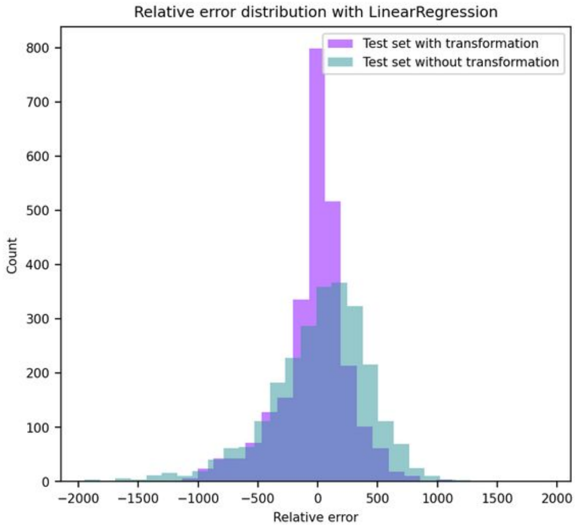
Standardize

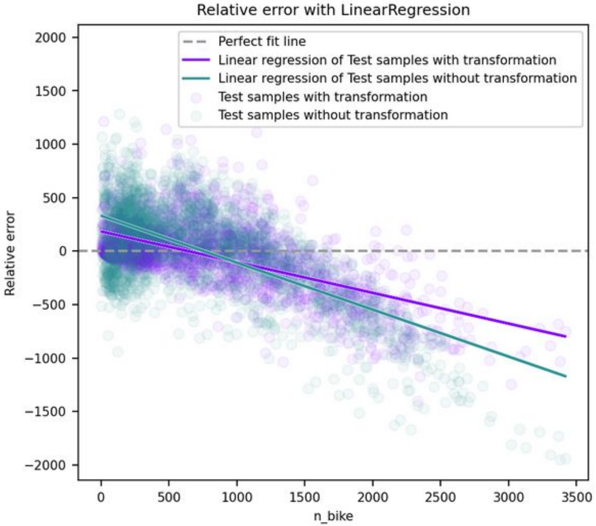
n_bike
temperature
wind
dew

Model Selection



Design **experiments** to select a statistical model from a set of candidate models.





Model Name: LinearRegression

Description: Ordinary Least Squares algorithm

Prevents Overfitting: no

Handles Outliers: no

Handles several features: no

Adaptive Regularization: no

Large Dataset: no

Non linear: no

Interpretability Score: 5 / 5

When to Use: Highly interpretable, no introduced bias

When to Use Expanded:

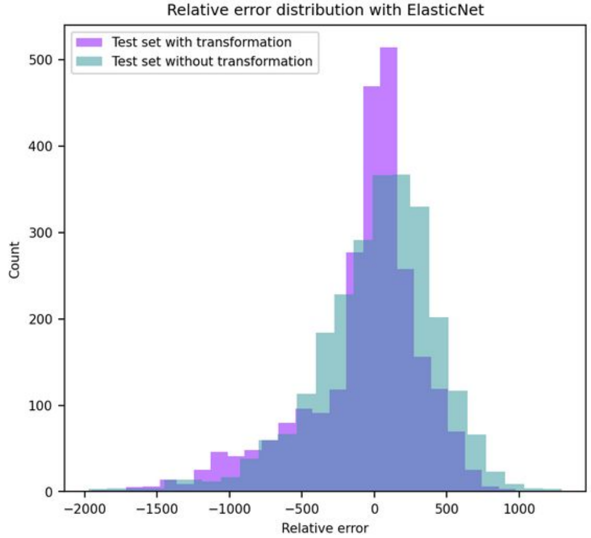
- Data consists of few outliers
- Little variance between output labels
- All of the input features are not only independent but also are not correlated.

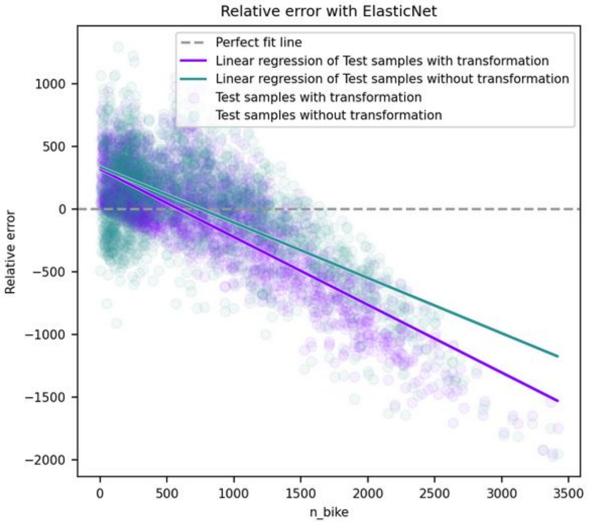
Advantages:

- Easy to interpret results
- Low complexity level

Disadvantages:

- At risk of multicolinearity if input features are correlated
- Small errors/outliers in target values can drastically impact model





Model Name: ElasticNet

Description: Ordinary Least Squares with both an L1 and L2 regularization term. The weights of the L1 vs. L2 regularization terms are controlled by an l1_ratio parameter.

Prevents Overfitting: yes

Handles Outliers: no

Handles several features: yes

Adaptive Regularization: no

Large Dataset: no

Non linear: no

Interpretability Score: 3 / 5

When to Use: Blend Ridge and Lasso

When to Use Expanded:

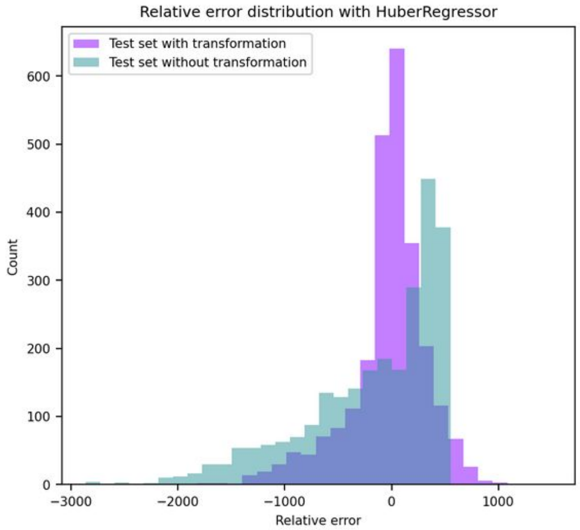
- Data consists of few outliers
- May be some correlation between input features
- Avoid overfitting
- Feature selection

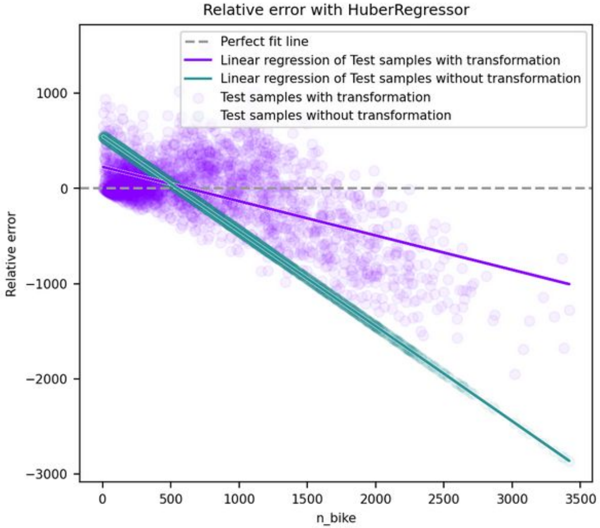
Advantages:

- Incorporate the feature selection abilities of Lasso with the regularization abilities of Ridge.

Disadvantages:

- In lowering variance, incorporates a degree of bias into the model.
- Can be difficult to tune alpha to attain a desirable balance between OLS and regularization terms
- Higher computational cost than Ridge or Lasso





Model Name: HuberRegressor

Description: A linear model designed to deal with outliers in the data and/or corrupted data. Does not ignore the outliers, but rather gives them a lower weight.

Prevents Overfitting: no

Handles Outliers: yes

Handles several features: no

Adaptive Regularization: no

Large Dataset: no

Non linear: no

Interpretability Score: 4 / 5

When to Use: Outliers and want quickest algorithm

When to Use Expanded:

- Want quick analyses of data ignoring outliers

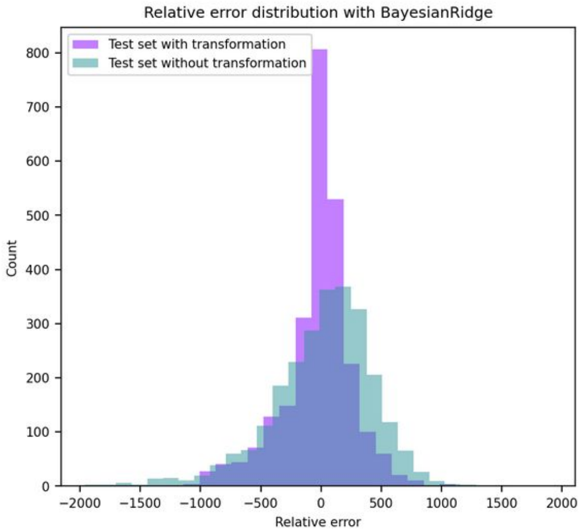
Advantages: '

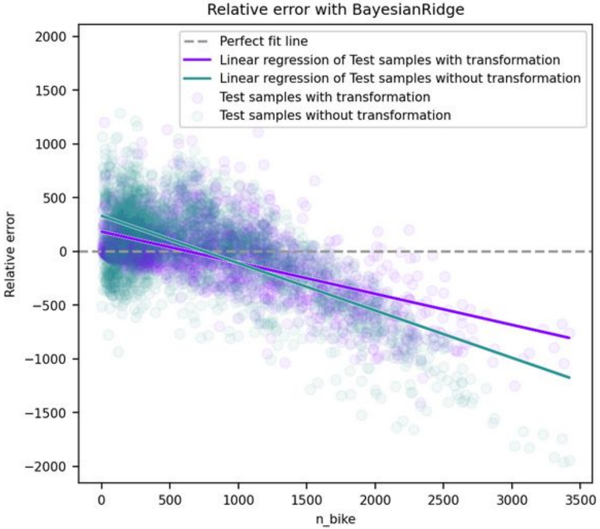
- Faster than RANSAC and TheilSen (as long as the number of samples is not too large)

- Does not completely ignore data points it deems as outliers

Disadvantages:

- Break down with large numbers of input features





Model Name: BayesianRidge

Description: Similar to Ridge but the regularization parameter is tuned to fit the data during the training process.

Prevents Overfitting: yes

Handles Outliers: no

Handles several features: no

Adaptive Regularization: yes

Large Dataset: no

Non linear: no

Interpretability Score: 2 / 5

When to Use: Ridge but don't want to set regularization constant

When to Use Expanded:

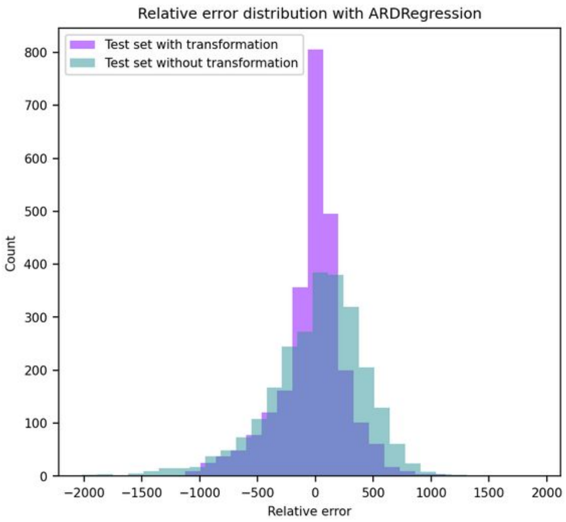
- Are seeking results similar to Ridge, but willing to sacrifice interpretability for time saved not having to test different regularization weights

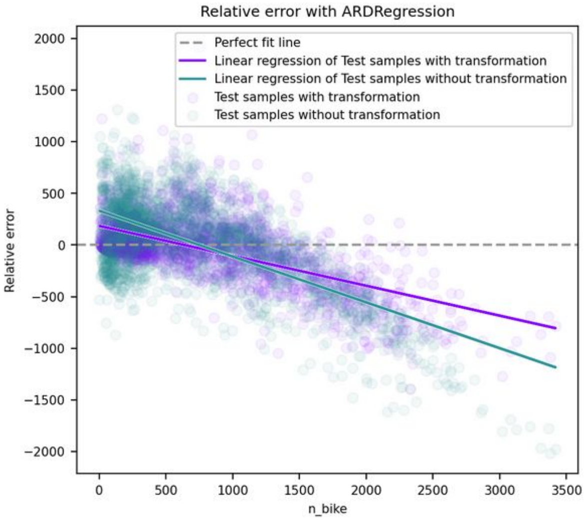
Advantages:

- No need to tune alpha value
- Adapts well to data on hand

Disadvantages:

- Less interpretable results





Model Name: ARDRegression

Description: BayesianRidge with sparser weight values. Almost like a version of BayesianLasso.

Prevents Overfitting: yes

Handles Outliers: no

Handles several features: yes

Adaptive Regularization: yes

Large Dataset: no

Non linear: no

Interpretability Score: 2 /5

When to Use: Lasso but don't want to set regularization constant

When to Use Expanded:

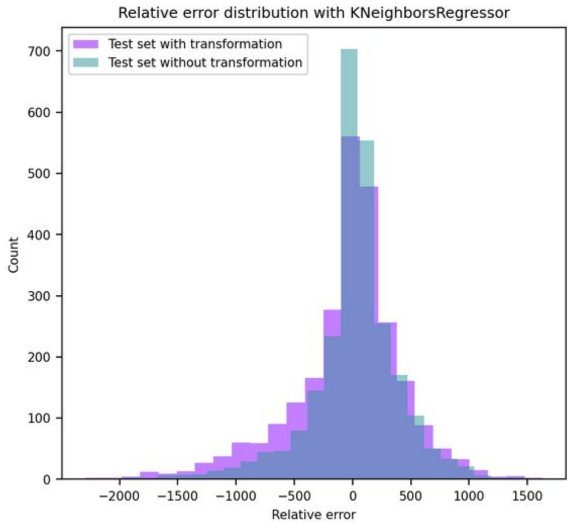
- Are seeking results similar to Lasso, but willing to sacrifice interpretability for time saved not having to test different regularization term weights

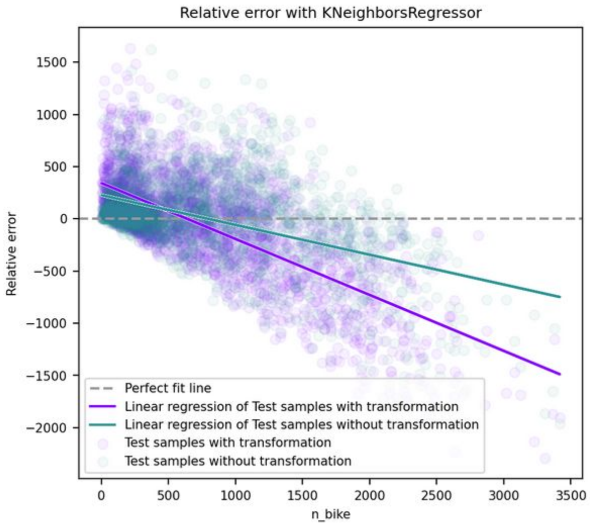
Advantages:

- No need to tune alpha value
- Adapts well to data on hand
- Reduces weight of unimportant features

Disadvantages: '

- Less interpretable results
- Computationally expensive (can't handle very large datasets)





Model Name: KNeighborsRegressor

Description: Creates a model based off of the k nearest neighbors at any given point. Where k is an input argument.

Prevents Overfitting: no

Handles Outliers: no

Handles several features: no

Adaptive Regularization: no

Large Dataset: no

Non linear: yes

Interpretability Score: 5 / 5

When to Use: Nonlinear data, interpretability is important, unimportant features

When to Use Expanded:

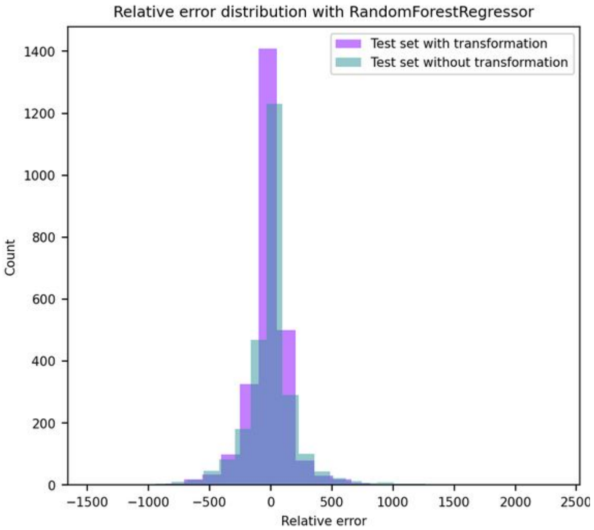
- When you are unsure of the structure of your data and want a model that will fit well
- Not concerned with overfitting
- Interpretability is important

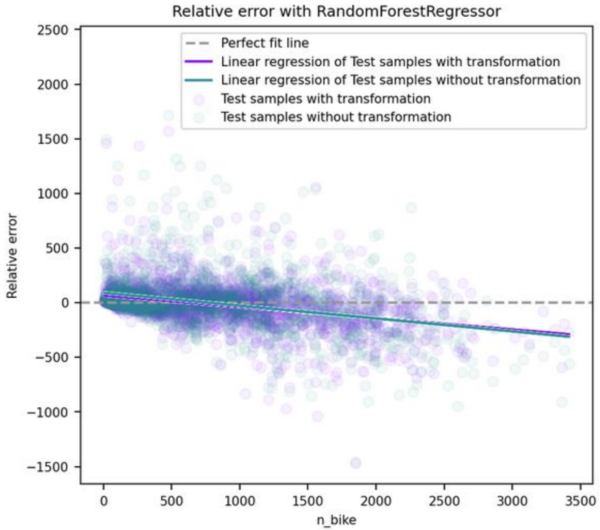
Advantages:

- Fits very well to data of various structures
- More interpretable than other nonlinear models

Disadvantages:

- Extremely impacted by outliers and corrupt data
- Need several more samples than features for quality results
- Difficulty dealing with large numbers of features





Model Name: RandomForestRegressor

Description: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Prevents Overfitting: yes

Handles Outliers: yes

Handles several features: yes

Adaptive Regularization: no

Large Dataset: yes

Non linear: yes

Interpretability Score: 3 / 5

When to Use: Nonlinear data groups in buckets

When to Use Expanded:

- Data is not linear and is composed more of "buckets"

- Number of samples > number of features

- There are dependent features in the input data. DTR handles these correlations well.

Advantages:

- Can export tree structure to see which features the tree is splitting on

- Handles sparse and correlated data well

- Able to tune the model to help with overfitting problem

Disadvantages:

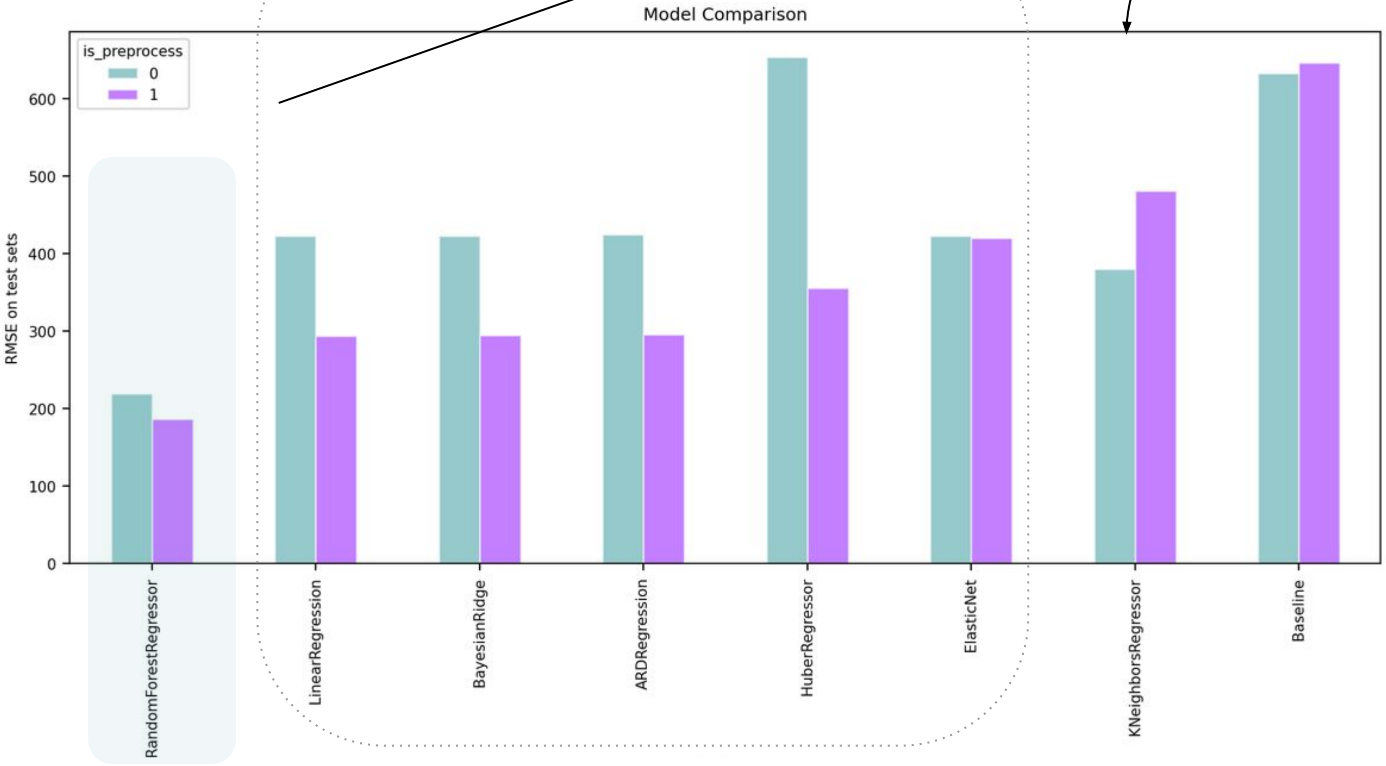
- Prediction accuracy on complex problems is usually inferior to gradient-boosted trees.

- A forest is less interpretable than a single decision tree.

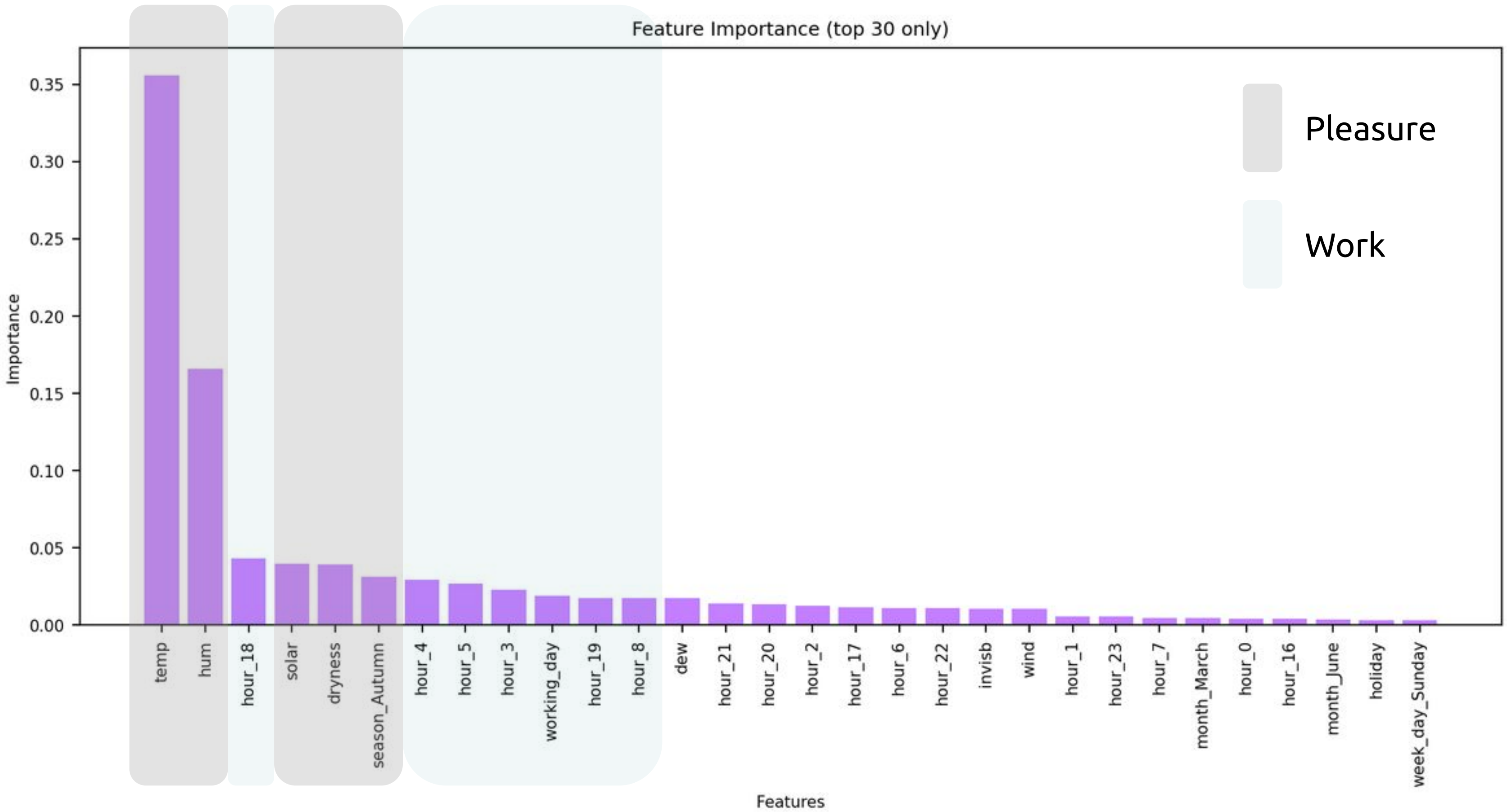
Comparison

Regularization → Worst results in **linear** models

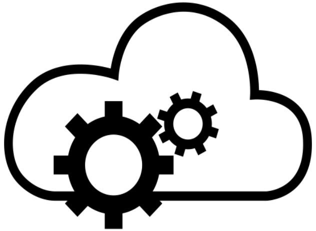
High **number of features** for KNR



RandomForestRegressor



Model Deployment



Integrate the selected model into an existing **production** and stable environment where a **client** request the model

Easy to use

Feature	Format
Date	dd/mm/yyyy
Hour	int
Temperature	°C
Humidity	%
Wind speed	m/s
Visibility	10m
Dew point temperature	°C
Solar Radiation	MJ/m2
Rainfal	mm
Snowfall	cm
Seasons	{"Winter", "Autumn", "Spring", "Summer"}
Holiday	{"Holiday", "No Holiday"}

```python
import requests
from preprocessing import preprocess

X = None  # Matrix of raw features
url = 'http://localhost:5000/predict'  # API request url

def serialize(df):
    return [[value for value in row] for row in df.values]

# preprocess de feature matrix
X = serialize(preprocess(X))
# request the API
r = requests.post(url, json={'inputs': serialize(X)})

# get the predictions
prediction = r.json()
```

Create
Feature Matrix

Request the API
< 10 lines of code