Bike Share Project

- Business Case

The hourly prediction of the bike sharing count value is not only important to estimate the expected revenue of the bike sharing service, but also to provide the required amount of bicycles at each station of a distributed bike sharing service. With more information about different stations, you could predict and schedule the rebalance of given back bikes. In this task, I investigate the prediction of the hourly bike count based on the specific hour, expected weather and day information

The following steps were performed to analyze the Bike Sharing Dataset and build a predictive model:

Descriptive Analysis
Missing Value Analysis
Outlier Analysis
Correlation Analysis
Model selection
Random Forest Training and Feature Ranking

Attribute Information:

hour.csv is the Data file

- instant: record index
- dteday : date
- season : season (1:winter, 2:spring, 3:summer, 4:fall)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- hr : hour (0 to 23)
- holiday : weather day is holiday or not
- weekday : day of the week
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
+ 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
- temp : Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
- atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-16, t_max=+50 (only in hourly scale)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

Numeric columns Analysis:

	temp	atemp	hum	windspeed
count	17379.000000	17379.000000	17379.000000	17379.000000
mean	0.496987	0.475775	0.627229	0.190098
std	0.192556	0.171850	0.192930	0.122340
min	0.020000	0.000000	0.000000	0.000000
25%	0.340000	0.333300	0.480000	0.104500
50%	0.500000	0.484800	0.630000	0.194000
75%	0.660000	0.621200	0.780000	0.253700
max	1.000000	1.000000	1.000000	0.850700

Categoric columns Analysis:

```
season has [1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4] values


holiday has [0, 1]
Categories (2, int64): [0, 1] values


mnth has [1, 2, 3, 4, 5, ..., 8, 9, 10, 11, 12]
Length: 12
Categories (12, int64): [1, 2, 3, 4, ..., 9, 10, 11, 12] values


hr has [0, 1, 2, 3, 4, ..., 19, 20, 21, 22, 23]
Length: 24
Categories (24, int64): [0, 1, 2, 3, ..., 20, 21, 22, 23] values


weekday has [6, 0, 1, 2, 3, 4, 5]
Categories (7, int64): [6, 0, 1, 2, 3, 4, 5] values


workingday has [0, 1]
Categories (2, int64): [0, 1] values


weathersit has [1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4] values
```

Analysis and data preparation:

Missing Values Analysis:

The missing value analysis revealed the data set does not contain any not-a-number or null values which require a replacement for further processing.

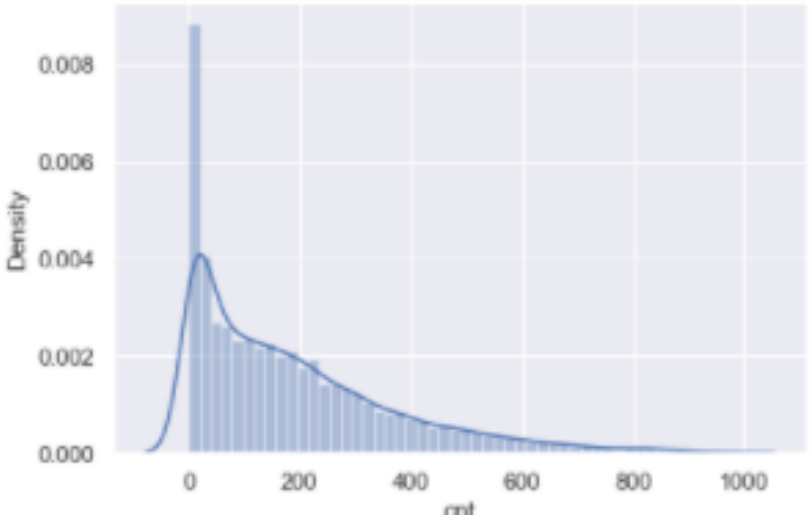
Feature Selection:

Instant , dteday, casual ,registered, atemp are the features not required for model building in the dataset.

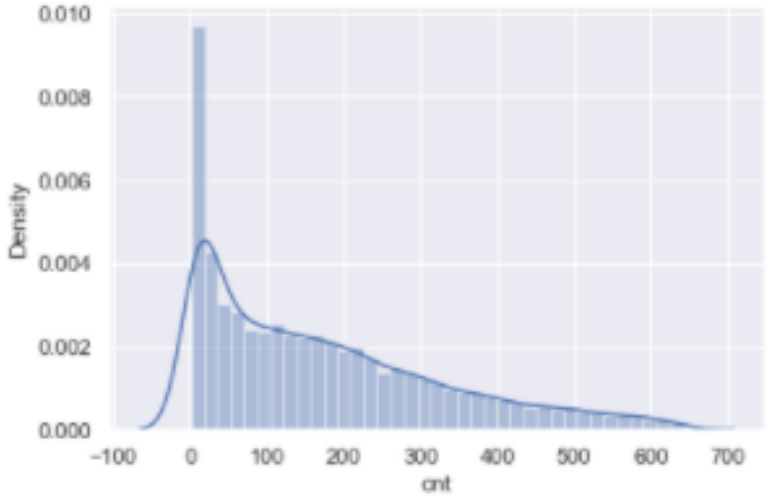
Outlier Analysis:

In the following step, the outliers of the count values were removed using median and interquartile range (IQR) as the count values do not fit a normal distribution. This reduces the data set from 15641 to 15179 samples.

Data with outliers



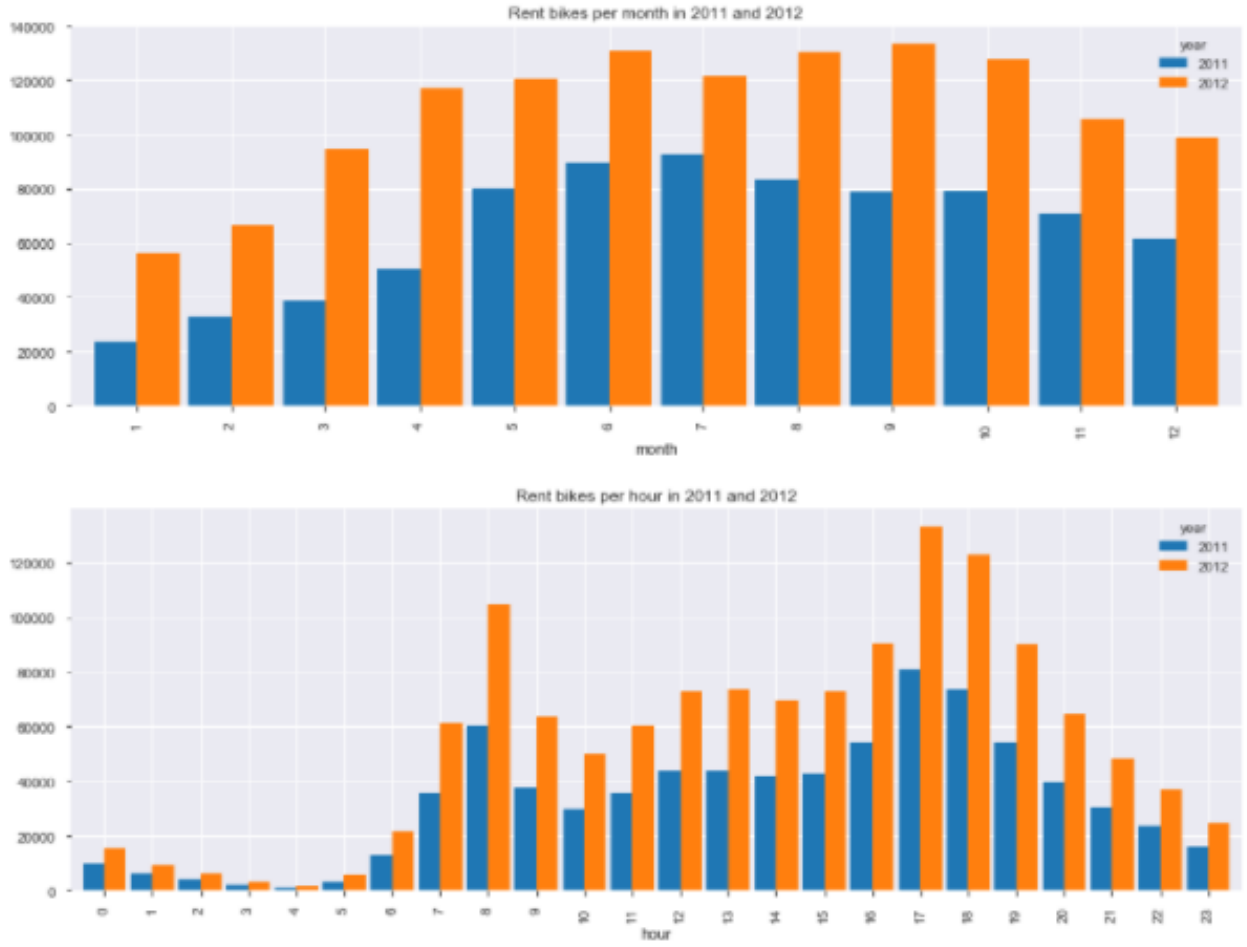
Data without outliers



Feature Importance for the model :

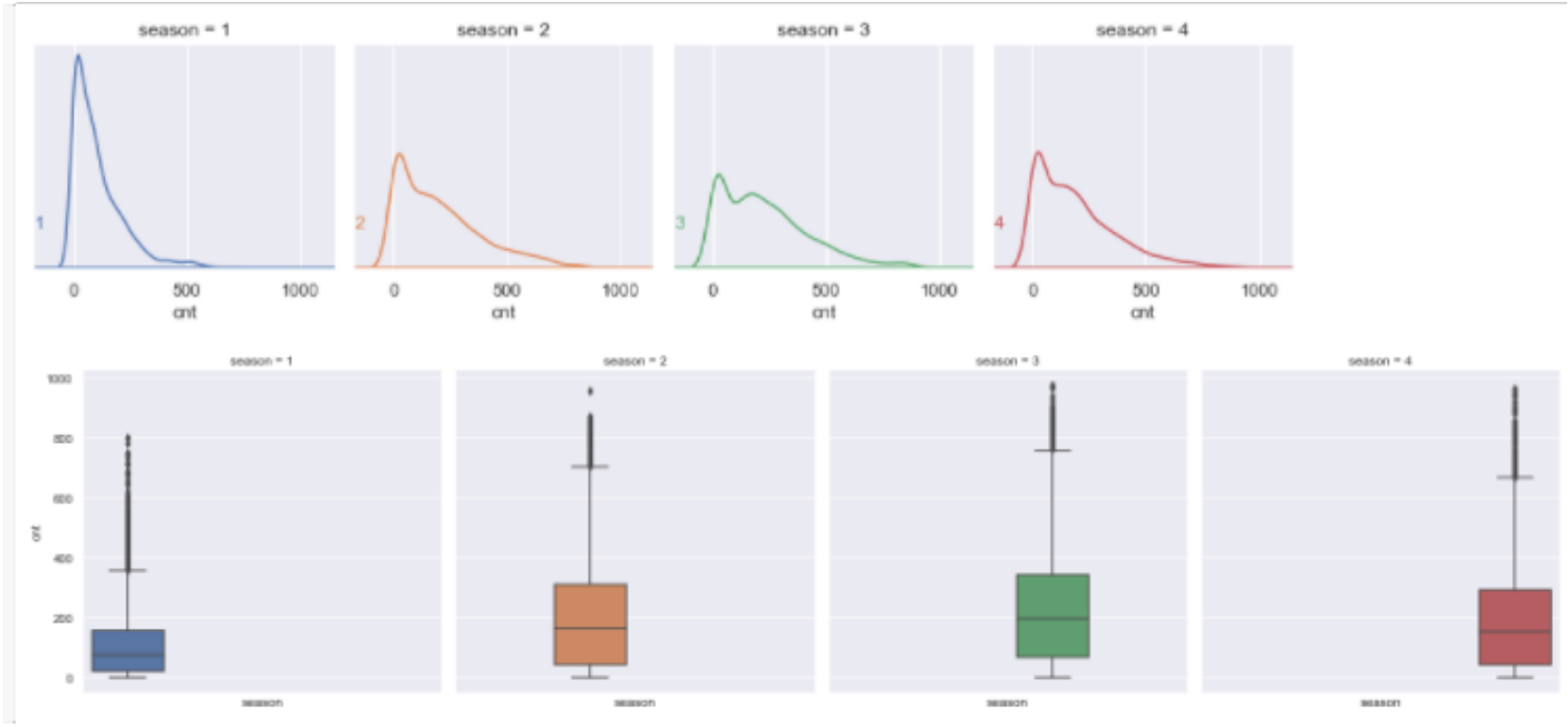
```
Feature ranking:
1. feature hr (0.604612)
2. feature temp (0.154340)
3. feature hum (0.062903)
4. feature workingday (0.042834)
5. feature windspeed (0.033177)
6. feature mnth (0.025870)
7. feature season (0.024970)
8. feature weathersit (0.024464)
9. feature weekday (0.023550)
10. feature holiday (0.003279)
```

Count plot w.r.t Month & hour

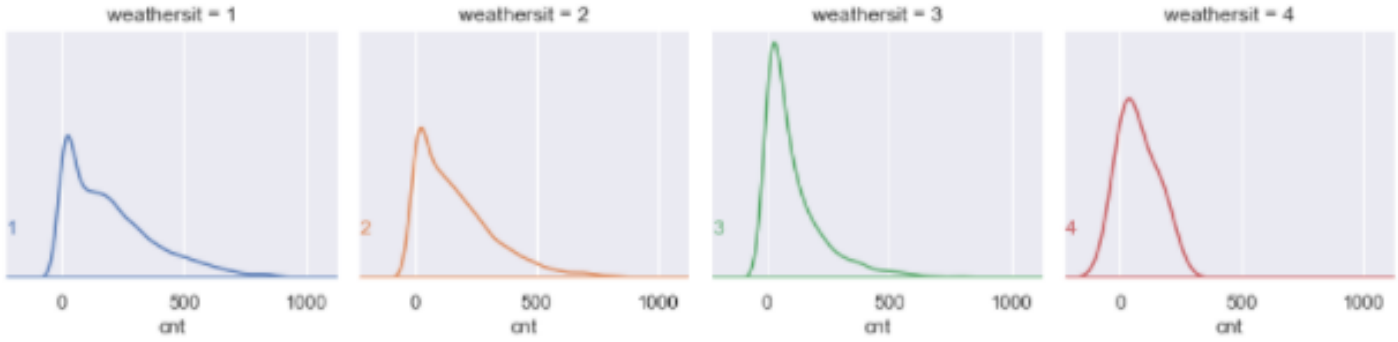


Interpretation: . The hourly box plots show a local maximum at 8 am and one at 5 pm which indicates that most users of the bicycle rental service use the bikes to get to work or school. Another important factor seems to be the temperature: higher temperatures lead to an increasing number of bike rents and lower temperatures not only decrease the average number of rents but also shows more outliers in the data.

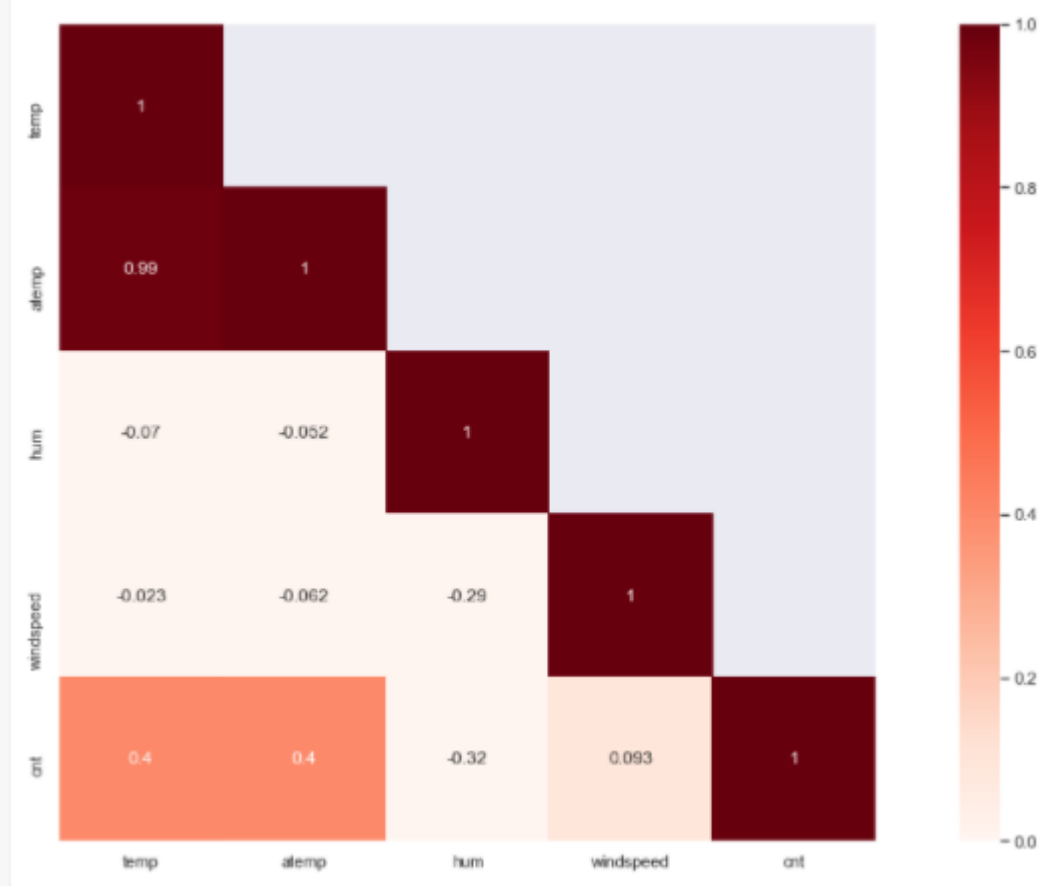
Bike sharing count w.r.t Season :



Bike sharing count w.r.t weather Situation:

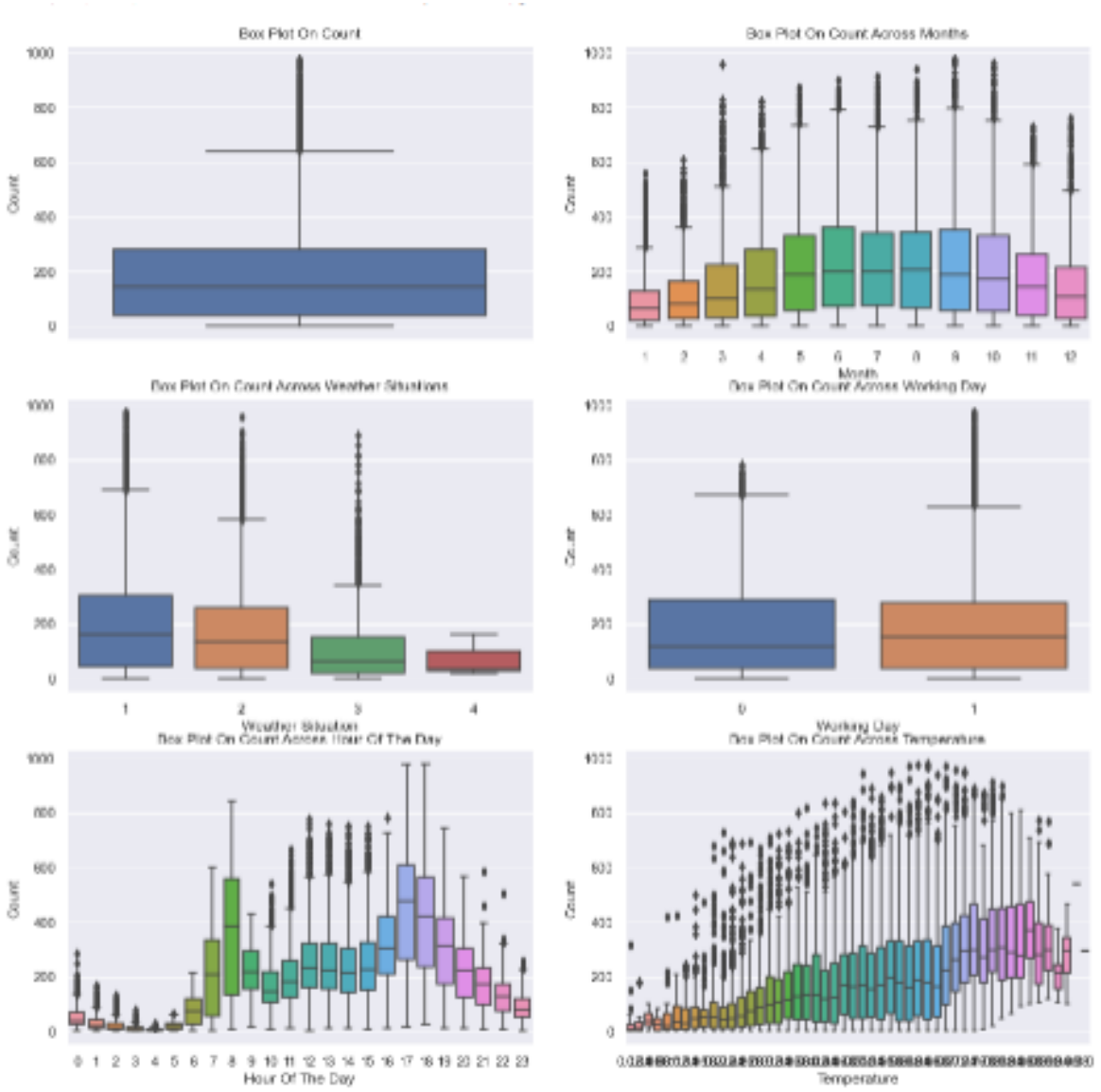


Correlation matrix for numerical features



The correlation analysis also revealed that temperature and feeling temperature are highly correlated. To reduce the model complexity and avoid collinearity, the feeling temperature features were dismissed.

The box plot says bike rents are more on weekdays than weekends or holidays



Model selection:

The prediction of the count values requires a regression algorithm based on categorical and numerical features. The dataset is quite small with less than 20K samples and the analysis steps revealed that a few features could be particularly significant. Based on these characteristics of the task and the data, I evaluated a set of possible algorithms: Lasso, Elastic Net, Support Vector Regression with different kernels, Ridge Regression and Random Forests.

By seeing below results it was observed that Random forest is giving the best performance. So we decided to go with Random Forest

Model	Mean Absolute Error	Mean Squared Error	R^2 score
SGDRegressor	41369.51	145.07	0.11
Lasso	35741.48	136.48	0.23
ElasticNet	44571.75	151.56	0.04
Ridge	35670.05	136.39	0.23
SVR	41581.06	141.76	0.10
SVR	28400.45	107.34	0.39
BaggingRegressor	5170.70	32.94	0.89
BaggingRegressor	34310.77	129.86	0.26
NuSVR	27506.76	110.56	0.41
RandomForestRegressor	4751.67	31.15	0.90
GradientBoostingRegressor	14605.97	77.18	0.69

Random Forest Model for final selection:

The random forests showed the most promising results on the Bike Sharing Dataset and were picked for the final result. The final random forest model consists of 200 decision trees trained on various-subsamples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. An internal needs at least four samples to split and the mean squared error was used to estimate the quality of a split.

The final model receives a mean absolute error of 16.84 & R2 value of 0.97

Model	Dataset	MSE	MAE	RMSLE	R^2 score
RandomForestRegressor	training	655.27	16.84	0.21	0.97
RandomForestRegressor	validation	4997.89	34.07	0.18	0.89

Hyper parameter Tuning:

Using Randomized searchcv we get the best Hyper parameters for better model performance .After implementing best hyperparameters for training the model ,results are slightly improved.

Model	Dataset	MSE	MAE	RMSLE	R^2 score
RandomForestRegressor	training	461.93	14.14	0.18	0.98
RandomForestRegressor	validation	4851.13	31.60	0.16	0.90

5 – Code

The python code to reproduce the provided analysis, plots and models is provided including documentation and unittests. This guarantees the reliability of the code securely, facilitates maintenance and allows potential colleagues to extend the code.

Analysis included in the below notebook
Bikesharingnotebook.ipynb

To execute the main file

python src/main.py

For tests
python -m unittest --verbose

Part 2 – Large-scale dataset

The Sklearn & Pandas packages work pretty well (small & medium datasets)on a single node system where data can fit into memory but with Terabyte data it cannot handle by single node system as data cannot be stored in main memory . This causes computationally expensive and sometimes will crash if we use large-scale datasets.

A good solution would be with Apache Spark for batch data or spark streaming for live streaming data where data can be distributed into chunks across cluster on multiple machines. Apache Spark ML is a machine learning framework highly optimized for distributed computation and would allow the utilization of a computation cluster. Spark ML can run on Hadoop, cloud managed services, Kubernetes, etc. , can access data from popular Apache databases like Apache Cassandra and claims to be 100 times faster than classic algorithms.

Problems :
If we designed Spark for a specific volume of data if we get more data once in while then again performance issues may arise. Instead we can do Finetuning or performance tuning by updating the configuration settings in spark. But if we see continuously increasing the data then we need to add more nodes in cluster which might be a bit expensive.

So to avoid this kind of problems we use lift & shift model to deploy our application in cloud and process it there (Eg: Data-proc in GCP) .

We can also use cloud platforms like (GCP,AWS,MS Azure)where there will be big clusters which will take care of the infrastructure and auto scaling by themselves.

I have theoretical knowledge on distributed computation and cluster architectures. I have no hands-on experience with frameworks like Hadoop or Apache Spark ML so far.