Bike Sharing Demand Prediction

Business Case:
The prediction of hourly bike count is vital for estimating revenue and ensuring an adequate supply of bicycles at each station. This information aids in efficiently managing bike distribution for a widespread sharing service.

Data Analysis and Preparation:
Upon careful examination, all columns exhibit complete data. Numeric features such as temperature, humidity, and windspeed have been appropriately normalized (Fig1).

The 'cnt' feature strongly correlates with both 'registered' and 'casual' (Fig2). Since 'temp' and 'atemp' exhibit identical correlation with 'cnt', a decision was made to drop 'temp' after conducting VIF analysis. Bike demand increases with the rise in humidity. Whereas an observation is made, there might be record missing for humidity, which results in a white line without any record in the graph(Fig 1).

Additionally, 'cnt' demonstrates a right-skewed distribution, prompting the removal of outliers using the IQR method. (Fig 3)

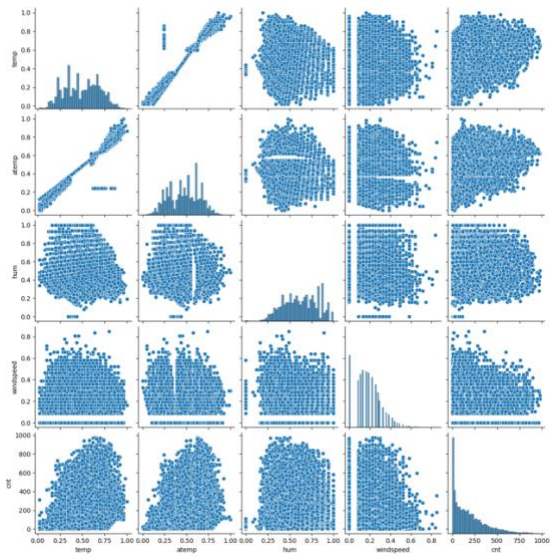

Fig 1

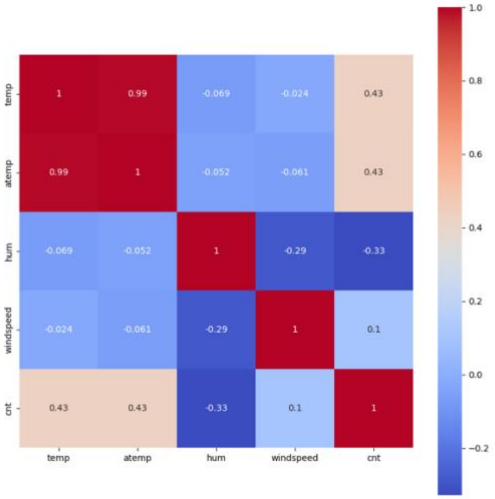

Fig 2

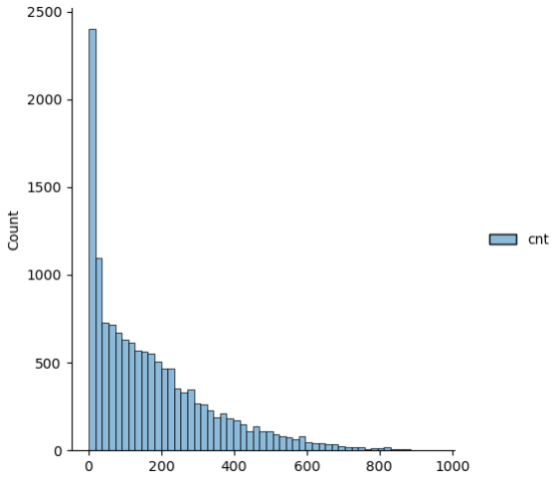

Fig 3

Upon in-depth analysis, distinct patterns emerged between working days and holidays. Regular working days saw a substantial surge in rentals compared to holidays, with peak demand at 8 AM and 5 PM, indicative of commuter usage. Conversely, peak demand shifted during holidays between

12 and 5 PM (Row 2 in Fig 4). During the summer season and under clear or partly cloudy weather conditions, demand experienced a significant increase.
Box plots gave better visuality presenting the demand sessions.

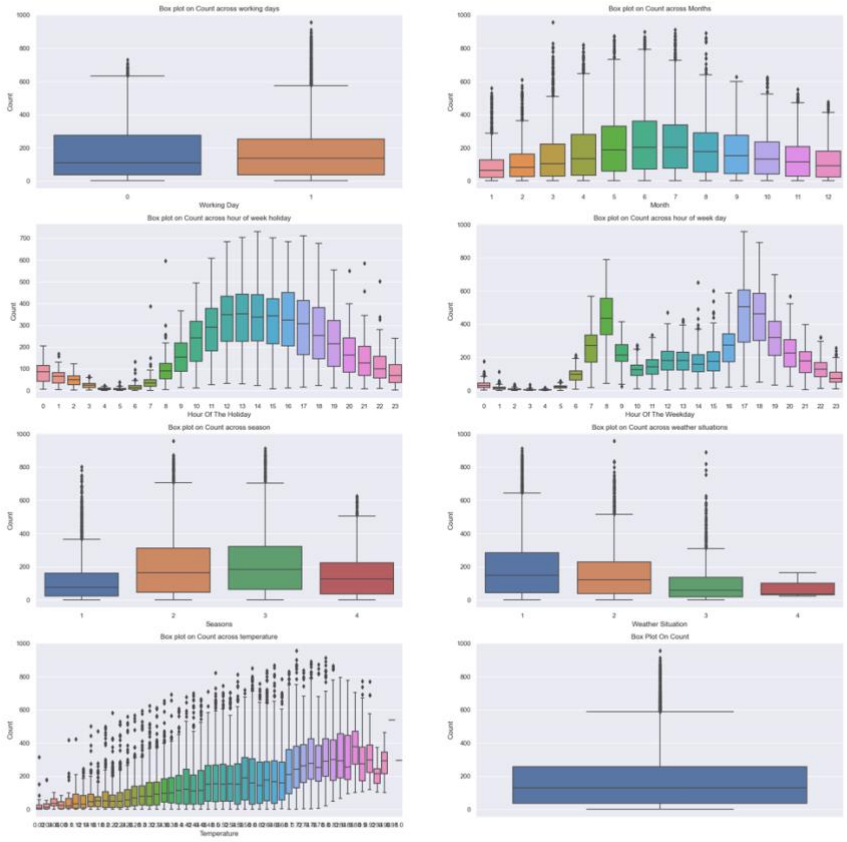

Fig 4

Model Selection:
Regression algorithms were assessed given the dataset's size and the importance of specific features. The evaluation included Ridge, Lasso, KNeighbour, Support Vector Regression, XGBRegressor and Random Forests. The XGBRegressor emerged as the most promising model.

XGBRegressor:
The XGBRegressor is an ensemble learning algorithm based on decision trees. Internally, it works by constructing an ensemble of decision trees, where each tree corrects the errors of its predecessor. The model is evaluated with metrics such as mean absolute error and R-squared.

The final model achieved, by hyperparameter tuning, a mean absolute error of 93, validated through 5-fold cross-validation. This signifies the average absolute difference between the predicted and actual bike counts.

This comprehensive analysis underscores the pivotal role of weather conditions and day type in predicting bicycle rental patterns.

SourceCode:
https://github.com/sanduluP/BikeDemandPrediction.git
The result can be reproducible, as random_state is used. *demandprediction.py* is the main file to be executed. Config file consists of user input details. *XGB_Regressor.py* file consists of model-related content.

- Considering that the code will be used in a production environment and maintained by colleagues, several best practices should be followed:
    - **Concise Code Documentation**: Readable code, detailed comments on methods, and explaining code logic help others understand
    - **Logging**: Implementation of logging ensures that events causing any issues can be identified and addressed promptly.
    - **Version Control**: Utilizing a version control system like Git allows for easy tracking of changes and collaborative development.
    - **Automated Unit Testing**: Ensuring unexpected break of functionalities doesn't happen.
- XGBRegressor is relatively fast to train, but its training time increases linearly with the increase in number of data points.
    - **Parallel model training:** Training models in parallel across clusters is a powerful technique for accelerating the model-building process. It takes advantage of distributed computing capabilities, reducing training time for complex models.
    - **Test at every level:** thorough testing at every level during development help identify code breakdown or data issue early on minimising the impact on downstream processes.
    - **Resource utilization:** For timely project completion, resources (time, budget) and memory allocation can be enhanced to meet expectations based on customer preference and project requirements.
- Overview of consideration:
    - **Project Stage Assessment:** Evaluating the current stage of a project is important. If it's already in the implementation stage, conducting a proof of concept with scaled data is essential to identify potential challenges and observation points.
    - **Pricing adjustment:** Scaling a project often involves additional costs. The sales team must propose a new pricing structure for the increased scope and resources required.
    - **Scaling properties:** Scaling impacts various aspects, including time consumption, resource allocation, data quality checks, and rigorous testing at every level. Automation and feature engineering for dimension reduction are key strategies.
    - **Limits and challenges:** Scaling infrastructure can be expensive, and ensuring data integrity in distributed computing environments is a significant challenge. Code integration and maintenance become critical tasks when adopting new technologies.
    - **Big data technology utilization:** Given the agreed-upon complexity and budget, leveraging big data technology can be a viable solution.

My combined theoretical knowledge of big data and practical experience from previous Test Data Management Consultant role equip me to address the complexities effectively. My experience in handling production data for test environments, including GDPR compliance, positions me well to navigate these scenarios.