YoungEverest — Bike Sharing Demand Updated Midpoint Report

---

1. Updated Dataset Description & Data Cleaning

The dataset used is the Bike Sharing Demand dataset from Kaggle. It contains 10,886 hourly observations, including:

- Weather variables (temperature, humidity, windspeed)
- Calendar indicators (season, holiday, workingday)
- Timestamp information
- Total bike rentals (count), which is our regression target

Data Cleaning steps performed

1. **Removed leakage features** (casual and registered) because they sum directly to the target count.
2. Parsed the datetime column and extracted: hour, weekday, month, and year
3. Verified there were **no missing values**.
4. Reintroduced important categorical variables (season, weather) and encoded them using **One-Hot Encoding**.
5. Applied **StandardScaler** for linear models only.
6. Ensured **train/validation/test split** uses stratification and a fixed random seed.

2. Exploratory Data Analysis (EDA)

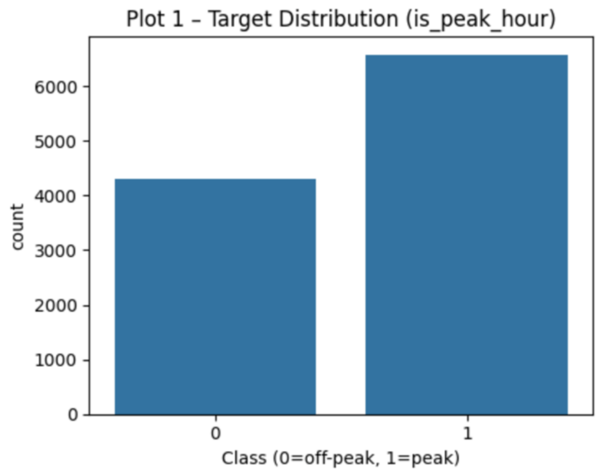
This section includes the two required midpoint plots.

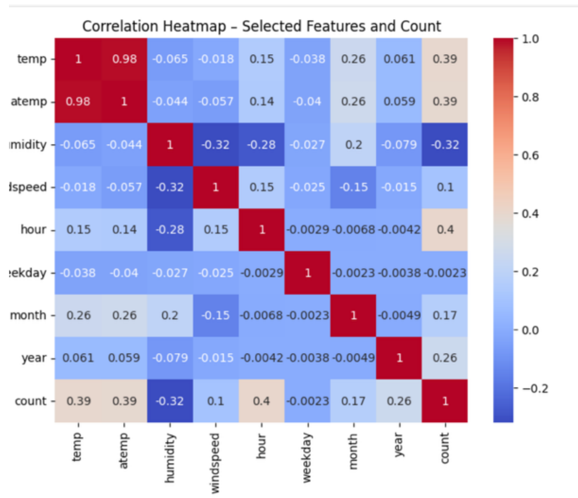
Plot 1: Target Distribution for Classification

Description:

A peak hour is defined as any hour where count ≥ 100.

This threshold is supported by the histogram and hourly usage pattern, where nighttime hours consistently fall below 100, and daytime/commute hours frequently exceed it.





Plot 2: Correlation Heatmap (Key Numeric Features)

Description:

Temperature and hour show strong positive correlation with bike rentals, while humidity shows a negative correlation. These relationships motivated feature choices for baseline models.

3. Train–Validation–Test Split

The dataset was split into:

- 70% Training set (7,620 rows)
- 15% Validation set (1,633 rows)
- 15% Test set (1,633 rows)

A **fixed random seed (42)** ensures full reproducibility.

4. Baseline Models with MLflow Tracking

We implemented four classical models:

Classification Baselines

1. Logistic Regression (scaled)
2. Decision Tree Classifier (unscaled)

Regression Baselines

1. Linear Regression (scaled)
2. Decision Tree Regressor (unscaled)

All models were logged using **MLflow**, including: parameters, metrics, model artifacts.

5. Results and Discussion

Table 1 — Classification Metrics (Validation + Test)

Model	Accuracy (Test)	F1 (Test)
Logistic Regression	**0.8953**	**0.9129**
Decision Tree Classifier	**0.8959**	**0.9154**

Summary:

The Decision Tree Classifier slightly outperforms Logistic Regression.

Tree-based models capture non-linear relationships such as:

- morning and evening commute peaks
- weather effects
- sudden changes in demand during extreme humidity

Table 2 — Regression Metrics (Validation + Test)

Model	MAE (Test)	RMSE (Test)
Linear Regression	**82.70**	**111.31**
Decision Tree Regressor	**51.52**	**84.63**

Summary:

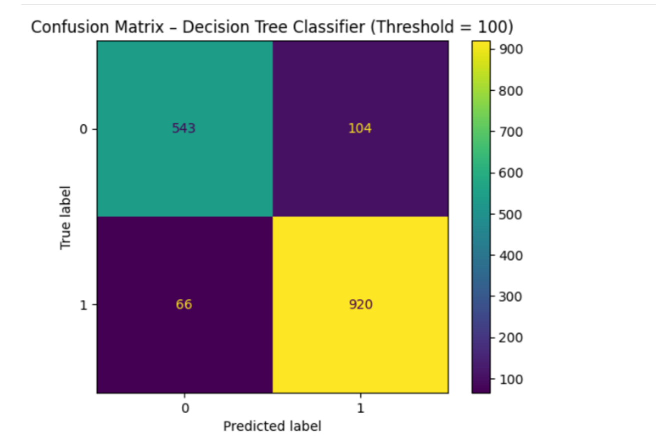
The Decision Tree Regressor significantly outperforms Linear Regression.

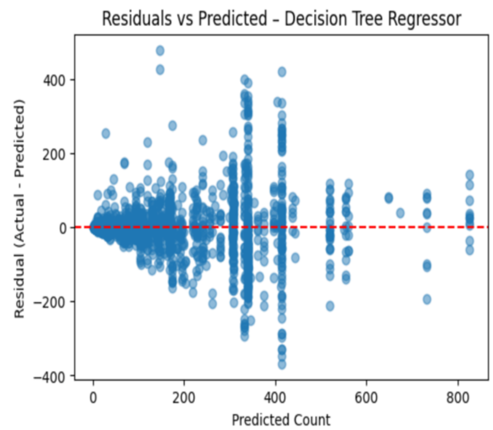
Linear Regression struggles with non-linearity, whereas trees adapt to sharp demand changes across hours and seasons.

6. Required Plots for Model Evaluation

Plot 3 — Confusion Matrix (Best Classifier: Decision Tree)

Shows strong classification performance with balanced predictions across peak and off-peak classes.





Plot 4 — Residuals vs Predicted (Best Regressor: Decision Tree)

Residuals are close to zero for moderate demand and show expected variance for extreme values.

7. Discussion: What Worked & Failure Modes

What Worked

- Decision trees captured complex interactions (weather × hour × temperature).
- One-hot encoding fixed earlier categorical processing issues.
- Scaling improved convergence for linear/logistic models.
- Stratified splits maintained class balance and improved stability.
- MLflow ensured complete reproducibility and proper experimental tracking.

Failure Modes / Limitations

- Linear models struggled with non-linear patterns and high variance.
- Residuals increase at higher rental counts (heteroscedasticity).
- Decision trees can overfit; deeper trees showed unstable validation metrics.

8. Neural Network Plan (For Final Report)

We will implement a **Multilayer Perceptron (MLP)** for both regression and classification:

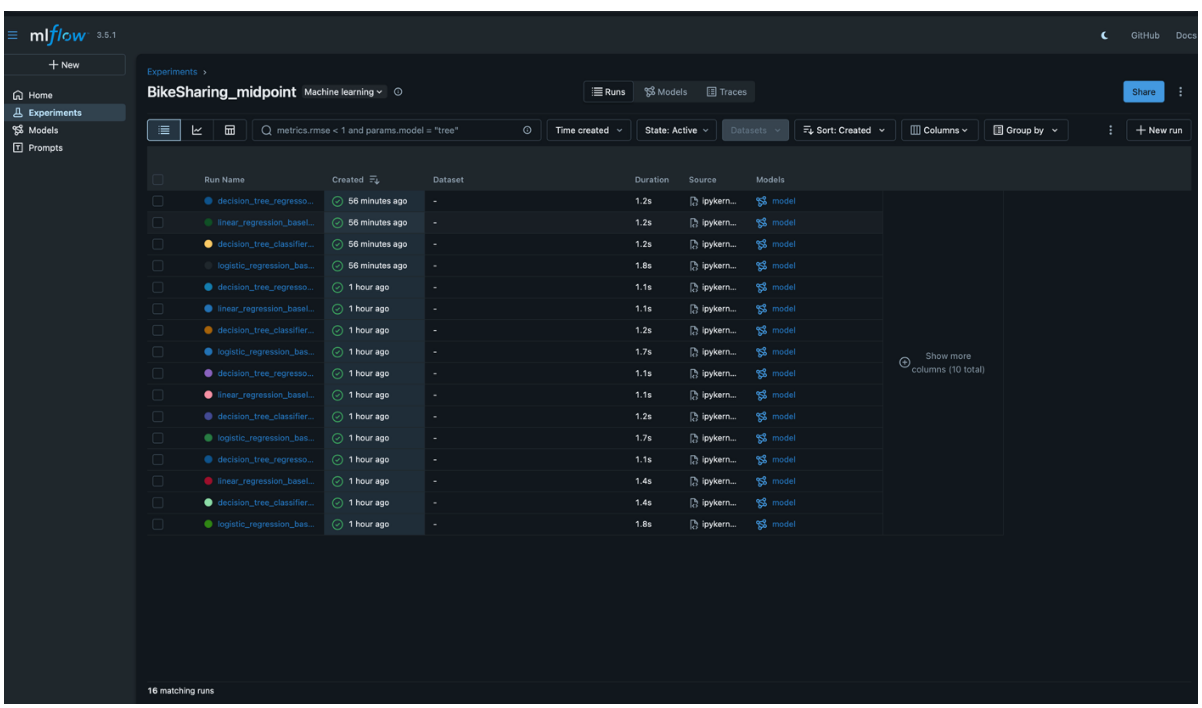
- 2–3 hidden layers (64–128 neurons each)
- ReLU activation
- Batch normalization
- Dropout for regularization
- Output layers:
  - Sigmoid for classification
  - Linear for regression
- Adam optimizer + early stopping

Justification:

MLPs handle non-linear relationships better than linear models and reduce the overfitting tendency of single decision trees.

9. Appendix — MLflow Experiment Tracking

This screenshot confirms that all baseline models were successfully logged with parameters, metrics, and artifacts, meeting the reproducibility requirement.



10. References

1. Fanaee-T, H., & Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2–3), 113–127.
2. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
3. MLflow Documentation. https://mlflow.org/
4. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), 90–95.
5. Waskom, M. L. (2021). Seaborn: Statistical Data Visualization. *Journal of Open Source Software*, 6(60), 3021.

GitHub Repository - https://github.com/arjun-sapkota887/Bike-Sharing-Demand