Smart Prediction for Shared Bicycles (Data Bootcamp Final Project) by Ray Xie

Introduction:

In recent years, there has been the rise of the shared bike programs as an environmental and trendy form of transportation within cities around the world. The way of commuting reduces traffic jam, carbon emissions, and improves users' health. So far, Washington D.C., New York City, and Beijing are some of the major cities that currently enjoy extensive bike-share systems with thousands of daily users.

One of the key questions for these sharing systems is: Can we accurately forecast the number of bike rentals on a given day? In my final report, I'm gonna focus on forecasting daily rental counts (cnt) from variables such as weather, season, and calendar variables. I believe accurate demand forecasts are required for planning operations because they can help city planners and service providers to allocate bikes efficiently, schedule maintenance, and adjust fleet size. For instance, miscalculating demand on a hot holiday can lead to empty stations and lost money, but overestimating on a rainy, cold workday can trigger unfilled bikes and additional operating expense.

Why this prediction is important to consider:

First, it enhances urban planning efforts by helping city officials anticipate transportation needs and promote sustainable alternatives to private-car use.

Second, it enables operational efficiency by allowing bike-sharing companies to adjust supply levels and maintenance schedules based on expected demand.

Lastly, predictive insights can support policy analysis, shedding light on how factors like holidays, seasonal changes, or traffic interventions influence public transportation system.

The content of this project:

My goal is to forecast daily bike rentals model using historical data from Washington D.C.'s bike-sharing system. The approach includes:

1. Performing exploratory data analysis to identify patterns.
2. Building and comparing Linear Regression, KNN, Random Forest, and Decision Tree models.
3. Evaluating model performance using RMSE and R square.
4. Interpreting results and next steps to understand key drivers of rental behavior.

Data Description:

This 2011-2012 dataset comes from the UCI Machine Learning Repository and initially collected by the Capital Bikeshare System in Washington D.C.. The data is aggregated on a daily basis and supplemented with weather and calendar information, making it a qualified resource for my predictive modeling.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     731 non-null    int64
 1   dteday      731 non-null    object
 2   season      731 non-null    int64
 3   yr          731 non-null    int64
 4   mnth        731 non-null    int64
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

```
[ ] # Confirm that 'cnt' is the sum of 'casual' and 'registered'
    # This validates the definition of our target variable
    (df['cnt'] == df['casual'] + df['registered']).all()
```

```
np.True_
```

```
[ ] # Check unique values for key categorical variables
    # This supports our earlier point that variables like 'season', 'holiday', and
    'weathersit' are categorical with discrete levels
    df['season'].unique(), df['weathersit'].unique(), df['holiday'].unique(), df
    ['workingday'].unique()
```

```
(array([1, 2, 3, 4]), array([2, 1, 3]), array([0, 1]), array([0, 1]))
```

```
# Validate normalized weather-related variables
# These should be between 0 and 1, as mentioned in the dataset description
df[['temp', 'atemp', 'hum', 'windspeed']].agg(['min', 'max'])
```

	temp	atemp	hum	windspeed
min	0.059130	0.079070	0.0000	0.022392
max	0.861667	0.840896	0.9725	0.507463

Analyzing the above 2 screenshots, the dataset contains 731 daily observations. Each row represents a single day and includes the target variable cnt as well as several predictors related to calendar events, weather conditions, and user behavior:

1. *Calendar-related variables* include the season (1=spring to 4=winter), year (0=2011, 1=2012), month, weekday, holiday status, and whether it was a working day.
2. *Weather-related variables* capture both categorical and continuous aspects of the environment, such as weathersit (ranging from 1 for clear days to 4 for extreme conditions), normalized temperature (temp), perceived temperature (atemp), humidity (hum), and wind speed (windspeed).
3. *User behavior variables* include the number of casual users (casual), registered users (registered), and their sum (cnt), which serves as the main prediction target.

Strength of this dataset:

Each variable has been carefully engineered. Specifically, temperatures are normalized using the min-max range for interpretability, and binary indicators like holiday or workday clearly define whether the day belongs to a weekend or holiday.

The dataset also captures human activity patterns. The separation of registered and casual users, for instance, allows researchers to distinguish between commuter usage and recreational use. On this project, I treat cnt as a whole to avoid data leakage.

Beyond that, there are no missing values in the data, which simplifies the preprocessing step.

Models and Methods:

Before introducing complex models, I first establish a baseline for comparison, which predicts the average number of daily rentals (cnt) for every observation. This naive approach provides a reference mean squared error that more advanced models must exceed.

```python
from sklearn.metrics import mean_squared_error
import numpy as np
# Calculate the baseline prediction: Use the mean of y
y = df['cnt']
baseline_preds = np.ones(len(y)) * y.mean()
mean_squared_error(y, baseline_preds)
```

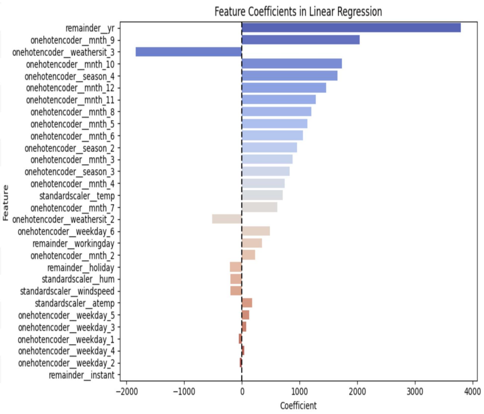
```
3747654.4350841474
```

The baseline model yields a high MSE of 3,747,654.44. This is expected as cnt typically ranges in thousands and MSE penalizes errors quadratically. Despite the model ignores all input features and daily variability, its poor performance makes it a valuable benchmark for assessing the effectiveness of more sophisticated models.

Linear Regression Model:

I begin with a linear regression model due to its simplicity, speed, and interpretability. It provides a useful baseline and helps reveal how features like temperature, season, and humidity relate to rental volume. Each coefficient offers clear insight into variable influence, making it ideal for building initial intuition. However, linear regression assumes linear relationships and can struggle with multicollinearity and outlier, which are limitations that more advanced models can address.

	Feature	Coefficient
27	remainder__yr	3794.500612
18	onehotencoder__mnth_9	2040.848736
10	onehotencoder__weathersit_3	-1836.823457
19	onehotencoder__mnth_10	1733.638485
2	onehotencoder__season_4	1653.510035
21	onehotencoder__mnth_12	1458.184375
20	onehotencoder__mnth_11	1282.551397
17	onehotencoder__mnth_8	1202.242852
14	onehotencoder__mnth_5	1132.563280
15	onehotencoder__mnth_6	1057.387895
0	onehotencoder__season_2	950.881027
12	onehotencoder__mnth_3	875.191225
1	onehotencoder__season_3	823.115645
13	onehotencoder__mnth_4	738.267791
22	standardscaler__temp	704.232010
16	onehotencoder__mnth_7	614.792461
9	onehotencoder__weathersit_2	-517.894601
8	onehotencoder__weekday_6	487.685064
29	remainder__workingday	351.616204
11	onehotencoder__mnth_2	226.675921
28	remainder__holiday	-206.796970
24	standardscaler__hum	-198.557775
25	standardscaler__windspeed	-198.297978
23	standardscaler__atemp	177.369216
7	onehotencoder__weekday_5	124.348659
5	onehotencoder__weekday_3	71.331145
3	onehotencoder__weekday_1	-57.845497
6	onehotencoder__weekday_4	41.808126
4	onehotencoder__weekday_2	-34.823198
26	remainder__instant	-4.956110



By looking at the linear regression coefficients, I got a clearer sense of what drives bike rentals:

As we expected, bad weather has the biggest negative impact. When the weather shifts to category 3 (light snow or heavy rain), rental volume drops by around 1800. That fits with real-life experience that fewer people want to ride bikes in unpleasant conditions.

On the flip side, summer (represented as season_4) boosts rentals by about 1600, and higher temperatures (standardized) add another 1285 rentals per standard deviation. Warm and sunny days clearly get more people out on bikes.

One result that stands out is July (mnth_7) posing a negative effect about -1210. That deviates from my view at first since I assume July would be a peak month. Maybe the reasons could be it’s too hot, or there’s less local commuting due to vacations.

Overall, the model picks up patterns that mostly match real-world expectations: good weather brings out the bikes, and bad weather keeps them away. It’s a simple model, but it already gives a helpful breakdown of what’s pushing rentals up or down.

```
RMSE: 792.03
MAE: 581.18
R²: 0.844
```

Performance Summary:

The linear regression model achieves an RMSE of 792.03, showing predictions are off by about 792 rentals on average. The MAE is 581.18, indicating that most errors fall within a few hundred bikes, which is considered to be a reasonable range. The R square score of 0.844 suggests the model explains about 84% of the variance in rental counts. While the model is not perfect, this is a relatively strong result for a simple model and shows it captures the majority of correct patterns. Thus, it is qualified to be as a solid baseline before moving on to advanced techniques.

KNN:

While linear regression offers a strong starting point and helped identify key predictors like weather and seasonality, it struggles with more complex relationships in the data. The biggest limitation is the assumption of linearity, which is something that rarely holds in real-world scenarios like our bike rentals case.

Let’s first analyze temperature variable. In real-life, people tend to avoid biking in both extreme heat and cold, which creates a U-shape preference curve that linear models can’t capture. Additionally, linear regression misses interactions among variables, like how humidity might affect ridership differently depending on the season.

To address these issues, I adopt the K-Nearest Neighbors model. Unlike linear regression, KNN makes no assumptions about the form of the relationship between features and the target. It simply finds the most similar days in the data based on inputs.

This flexibility makes KNN well-suited for capturing non-linear trends and hidden interactions. It can naturally learn that rentals drop off at both ends of the temperature scale without needing to model that curve explicitly.

Coding Method:

I start by splitting the data into 80% training and 20% testing sets using train test split, ensuring that evaluation is based on unseen data. Categorical features (season, month, and weather) are encoded, while numerical features (temperature and windspeed) are standardized using standard scale. These preprocessing steps are organized using a ColumnTransformer.

Then, I build a pipeline that combines both preprocessing and the K-Nearest Neighbors regressor. To further tune the model, I run a grid search over different values of n_neighbors (ranging from 5 to 50) with 5-fold cross-validation.

```
Train MSE: 490700.69
Test MSE: 631693.43
Train MAE: 493.39
Test MAE: 559.40
R²: 0.842
```

	Feature	Importance
0	instant	0.376159
9	atemp	0.115783
8	temp	0.115315
10	hum	0.074849
11	windspeed	0.034828
7	weathersit	0.020172
6	workingday	0.010614
3	mnth	0.004569
2	yr	0.004564
4	holiday	0.001287
1	season	0.000825
5	weekday	-0.001607

Performance Summary:

To interpret what drives KNN's predictions, I use the permutation importance, which measures how much model performance drops when each feature is randomly shuffled. The bigger the drop, the more important the feature.

Instant emerges as the top predictor that likely captures time-related patterns or seasonality not explicitly encoded elsewhere. Features like atemp and temp also ranked high, consistent with linear regression, highlighting that people tend to ride more in comfortable weather. Meanwhile, variables like humidity, windspeed, and weather conditions had moderate influence, while weekday, holiday, and season contributed little or even slightly negatively, indicating KNN didn't rely much on them for prediction.

In terms of performance, KNN clearly outperformed both the baseline (MSE > 3.7 million) and linear regression. Its test MSE dropped to about 631,693, and the MAE was 559.40, meaning predictions were typically off by around 559 rentals, a number that is much lower than linear regression's error. The R square score reached 0.842, showing that KNN explained over 84% of the variance in rental counts. Overall, KNN captured nonlinear relationships that linear regression missed, and it did so without making assumptions about the underlying data structure.

Random Forest Model:

While KNN offers a clear improvement over linear regression, it comes with notable limitations. One key issue is that it treats all neighbors equally. This can lead to overly smooth or noisy predictions, especially in dense regions where small variations in inputs are averaged together. For example, extremely cold or rainy days often lead to sharp drops in bike rentals. But KNN might blur these patterns by averaging them with nearby days, producing overly optimistic predictions. Additionally, KNN struggles in high-dimensional spaces, where irrelevant features can distort the notion of "closeness" and weaken prediction accuracy.

To address these challenges, I turn to a more powerful model: Random Forest. Like KNN, it captures nonlinear relationships, but it does so by building an ensemble of decision trees trained on random subsets of the data and features. This randomness helps the model generalize better and reduces overfitting. It also offers reliable feature importance scores, giving clearer insight into what truly drives bike rental behavior.

Coding Method:

To ensure fair comparison, I use the same 80/20 train test split and similar preprocessing steps: categorical features were one-hot encoded, and numerical features were standardized. These transformations were combined with a RandomForestRegressor in a pipeline.

I then applied GridSearchCV with 5-fold cross-validation to tune the number of trees. This setup helps prevent overfitting, ensures robust performance, and keeps the workflow clean.

Train MSE: 67,118.67
Test MSE: 487,935.09
Train MAE: 176.49
Test MAE: 449.30
R^2: 0.878

	Feature	Importance
25	remainder__instant	0.628732
26	remainder__temp	0.122663
27	remainder__atemp	0.112990
28	remainder__hum	0.058591
29	remainder__windspeed	0.030603
24	onehotencoder__weathersit_3	0.010110
23	onehotencoder__weathersit_2	0.006030
22	onehotencoder__workingday_1	0.005043
21	onehotencoder__weekday_6	0.003528
15	onehotencoder__holiday_1	0.003013
11	onehotencoder__mnth_9	0.001912
2	onehotencoder__season_4	0.001807
12	onehotencoder__mnth_10	0.001773
16	onehotencoder__weekday_1	0.001597
19	onehotencoder__weekday_4	0.001518
20	onehotencoder__weekday_5	0.001205
18	onehotencoder__weekday_3	0.001050
14	onehotencoder__mnth_12	0.001010
17	onehotencoder__weekday_2	0.000936
8	onehotencoder__mnth_6	0.000907
6	onehotencoder__mnth_4	0.000895
0	onehotencoder__season_2	0.000821
10	onehotencoder__mnth_8	0.000590
7	onehotencoder__mnth_5	0.000523
1	onehotencoder__season_3	0.000426
3	onehotencoder__yr_1	0.000414
4	onehotencoder__mnth_2	0.000359
13	onehotencoder__mnth_11	0.000334
5	onehotencoder__mnth_3	0.000314
9	onehotencoder__mnth_7	0.000308

Performance Summary:

To evaluate model performance, I analyze three metrics:

1. MSE captures average squared error. In our case, a test MSE of 488,000 is a major drop from the baseline (3.7 million) and linear regression (1.9 million), showing much better accuracy.
2. Test MAE reflects average prediction error. At 449, it means predictions are off by about 449 rentals per day.
3. R square indicates how much variance the model explains. A score of 0.878 means Random Forest accounts for nearly 88% of rental fluctuations, far stronger than the linear model (0.53).

Feature importance analysis shows that instant is by far the most influential variable. This is followed by temp and atemp, reinforcing earlier findings from KNN and linear regression's prediction that people prefer to biking more in moderate, comfortable weather. Variables like humidity and windspeed also contribute while calendar-based features such as individual months, weekdays, season, or holiday show minimal importance. This suggests that Random Forest focuses more on real-time numeric signals rather than static or repetitive categories.

Despite its strong performance, Random Forest comes with trade-offs. This model makes it difficult to interpret how specific variables influence individual predictions. It also tends to smooth out changes, such as sudden drops in ridership due to extreme weather on a holiday, by averaging across trees. As a result, it may underestimate rare but impactful events. Moreover, when uncommon combinations of inputs appear (an unusually hot Monday holiday in spring), the model might misrepresent them due to limited exposure in the training set.

Decision Tree Model:

Unlike Random Forest, Decision Tree offers full transparency as showing exactly how predictions are made at each split. While it's more prone to overfitting, this can actually be useful for exploring sharp behavioral shifts or rare cases in the data. For this special step, the goal isn't for optimal performance, but interpretability to see how the tree structures decisions and which features it relies on when predictions aren't averaged across models.

Coding Method:

In the modeling step, I train a Decision Tree Regressor using the same 80/20 train-test split and pipeline structure to ensure consistency across models. A grid search with 5-fold cross-validation was used to tune key parameters: max_depth, min_samples_split, and min_samples_leaf to minimize Mean Squared Error.

```
Train MSE: 219,564.76
Test MSE: 601,624.84
Train MAE: 329.04
Test MAE: 536.93
R²: 0.850
```

	Feature	Importance
25	remainder__instant	0.628732
26	remainder__temp	0.122663
27	remainder__atemp	0.112990
28	remainder__hum	0.058591
29	remainder__windspeed	0.030603
24	onehotencoder__weathersit_3	0.010110
23	onehotencoder__weathersit_2	0.006030
22	onehotencoder__workingday_1	0.005043
21	onehotencoder__weekday_6	0.003528
15	onehotencoder__holiday_1	0.003013
11	onehotencoder__mnth_9	0.001912
2	onehotencoder__season_4	0.001807
12	onehotencoder__mnth_10	0.001773
16	onehotencoder__weekday_1	0.001597
19	onehotencoder__weekday_4	0.001518
20	onehotencoder__weekday_5	0.001205
18	onehotencoder__weekday_3	0.001050
14	onehotencoder__mnth_12	0.001010
17	onehotencoder__weekday_2	0.000936
8	onehotencoder__mnth_6	0.000907
6	onehotencoder__mnth_4	0.000895
0	onehotencoder__season_2	0.000821
10	onehotencoder__mnth_8	0.000590
7	onehotencoder__mnth_5	0.000523
1	onehotencoder__season_3	0.000426
3	onehotencoder__yr_1	0.000414
4	onehotencoder__mnth_2	0.000359
13	onehotencoder__mnth_11	0.000334
5	onehotencoder__mnth_3	0.000314
9	onehotencoder__mnth_7	0.000308

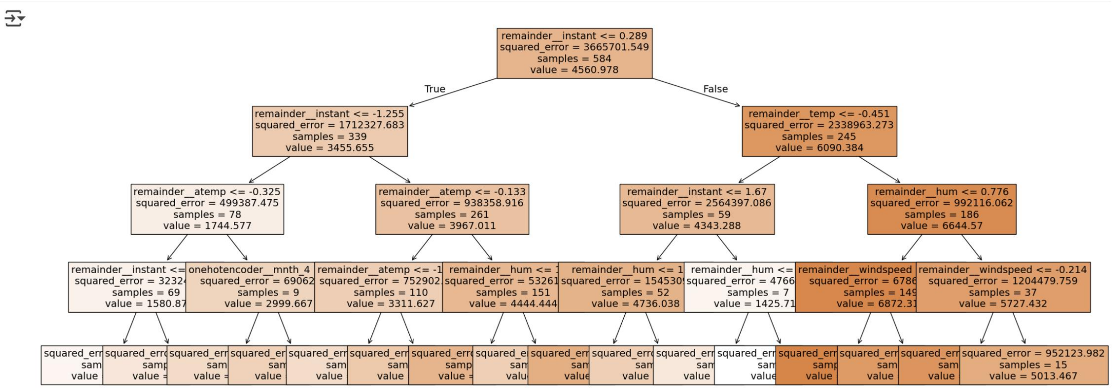
Performance Summary:

This model achieves a test MSE of 601,625 and MAE of roughly 537, meaning predictions are, on average, off by about 537 rentals. The R square of 0.850 shows that the model explains 85% of the variance in rental counts, which slightly behind Random Forest (0.878).

Compared to previous models, the decision tree performs quite well, especially given its simplicity. The gap between training and test errors suggests some overfitting issue, but not severe. More importantly, it offers full transparency. And we can see exactly how features drive predictions at each split. While less accurate than Random Forest, this model is helpful for understanding rental patterns.

Then by analyzing the feature importance, the pattern reinforces previous findings: Instant dominates, contributing over 60% of the model's predictive power. This highlights how powerful time progression is in capturing underlying trends, whether related to growing user adoption or platform evolution.

Following that, temp and atemp, representing actual and perceived temperature, together account for about 24% of total importance. Their role is consistent and intuitive: warmer days clearly drive more biking activity. On the other hand, categorical variables like weekday, holiday, and even weather show minimal impact, most scoring below 0.002. This suggests that once broader weather conditions and temporal trends are captured, these fine-grained calendar features add little incremental value. In this case, my decision tree confirms a clear hierarchy: bike rentals are driven primarily by time trends and weather comfort, while specifics like the day of the week or month play a secondary role.



Decision Tree Structure Interpretation:

I analyze the structure of the decision tree from top to bottom. At the top, the first split is based on remainder__instant, which is an index variable that likely captures latent time trends such as platform growth or seasonal cycles. Its dominance here aligns with earlier feature importance rankings.

On the left side, the tree focuses on colder or earlier days, splitting on atemp, hum, and even calendar features like mnth_4. For instance, when atemp smaller than or equals to -0.325, predicted rentals drop to around 1,745, revealing the consistent with lower demand during uncomfortable weather. This side of the tree is deeper, reflecting the model's need for more rules to handle variability under less favorable conditions.

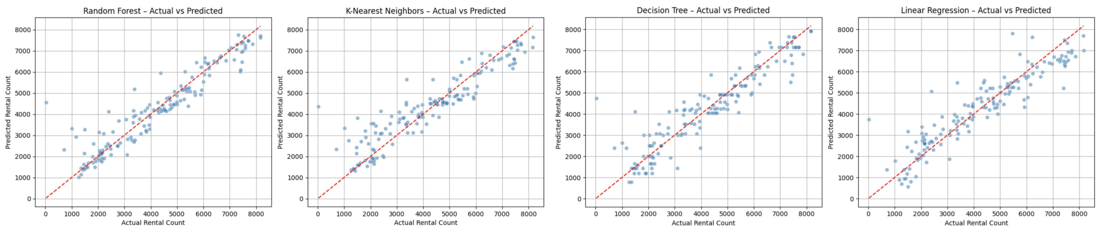
On the right side, the splits center around temp, hum, and windspeed, covering warmer days where bike usage is more predictable. These branches are shallower and converge to high rental predictions, highlighting the model's confidence under ideal conditions.

All in all, the structure of the tree tells us that: bike usage on colder, less predictable days requires more nuanced decision paths. While warm, dry conditions lead to stable, high rentals with fewer splits.

Result & Interpretation:

To better understand how the four models differ in structures and behaviors, I created 4 visual comparisons that highlights each of their characteristics.

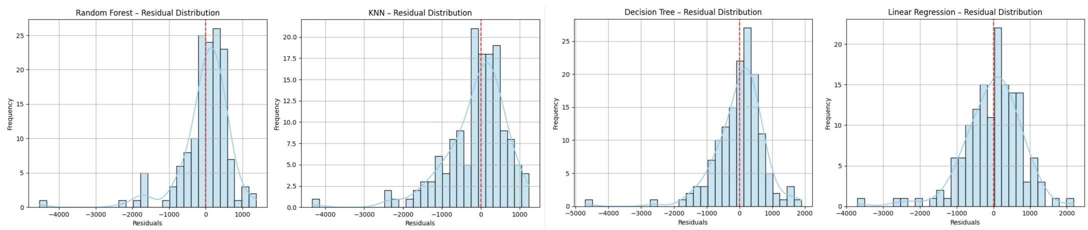
Actual vs Predicted Graph:



The scatter plots show how closely each model's predictions align with actual rental counts:

1. Random Forest produces the tightest fit along the diagonal, indicating accurate and balanced predictions across all demand levels.
2. KNN shows increasing spread as rental counts rise, suggesting it struggles with high-demand days.
3. Decision Tree performs better than KNN, though still shows some noise due to its segmented prediction structure.
4. Linear Regression deviates the most from the diagonal, underestimating high values and overestimating lows.

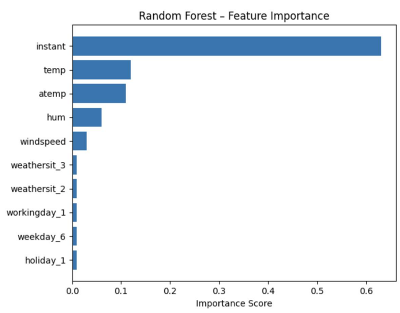
Histogram of Residuals:

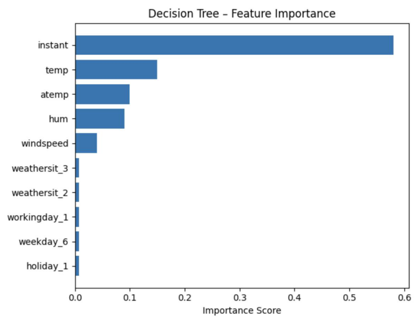


These plots show how each model's prediction errors are distributed: closer to zero indicates better accuracy and stability.

1. Random Forest: Residuals are tightly concentrated with a sharp peak near zero and thin tails, showing high accuracy and few extreme errors. Slight right skew, but overall very consistent.
2. KNN: Broader spread and slight left skew suggest less stability. Errors are more variable, especially on outlier days.
3. Decision Tree: Symmetrical and relatively compact residuals. Moderate variance, supporting its solid but not top-tier performance.
4. Linear Regression: Classic bell shape, centered at zero, but with fatter tails. It captures the average trend but struggles with high-variance or extreme cases due to its linear constraints.

Feature Importance Graph:



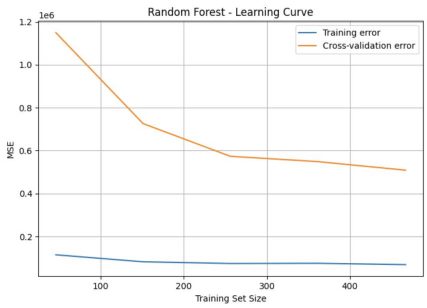


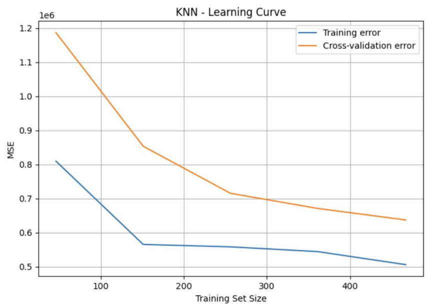
KNN and Linear Regression don’t have this feature importance scores because KNN doesn’t learn weights, and linear model relies on coefficients that aren’t directly comparable unless features are standardized.

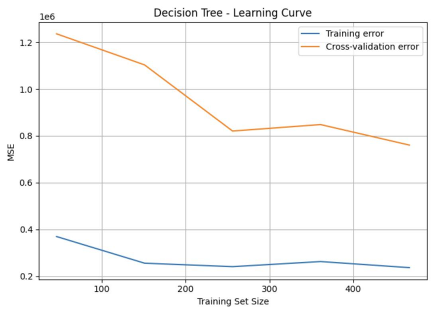
In both Random Forest and Decision Tree, the feature Instant dominates. As a time index, it captures hidden trends like platform growth or seasonal cycles. However, its overwhelming influence in Random Forest may indicate some reliance on sequential patterns.

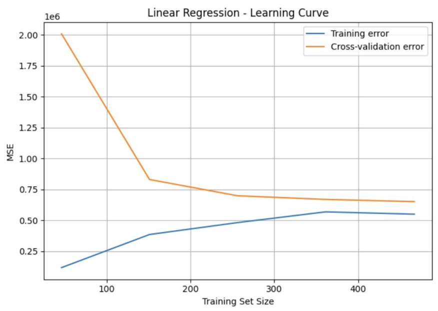
Weather-related features like temp, atemp, and hum rank next in both models, reflecting their secondary but consistent role across all methods. The Decision Tree model gives us a slightly more balanced perspective to weight these variables, suggesting it may generalize temporal effects less aggressively.

From my extra knowledge: Learning Plot









The learning curves illustrate how each model adapts to more training data and handles generalization:

1. Random Forest shows consistently low training error and declining validation error, indicating a low bias and moderate variance.
2. KNN has higher training error and a wide gap between training and validation curves. The

slight improvement with more data suggests high variance and limited capacity to generalize from local patterns.

3. Decision Tree exhibits classic overfitting: near-zero training error but stubbornly high validation error. The model memorizes training data well but fails to improve with scale.
4. Linear Regression reflects high bias. Training error increases with more data, and validation error stays flat and high. The small gap between the two indicates underfitting issue.

Conclusion and Next Steps:

Overall, my project compared four predictive models (Linear Regression, KNN, Decision Tree, and Random Forest) to forecast daily bike rental counts using the unbiased Bike Sharing dataset. Through a comprehensive analysis that included evaluation metrics, residual plots, feature importance, and learning curves, each model revealed distinct strengths and weaknesses.

Which Model Performs The Best?

Among the above 4 models, Random Forest consistently delivers the best overall performance. It achieves the lowest test MSE, the highest R square score (0.878), and produces residuals that are tightly clustered around zero. Its learning curve shows a steady improvement with more data, suggesting strong generalization ability and low variance.

Decision Tree and KNN also capture non-linear patterns but suffer from higher variance and less stability, as seen in their noisier residuals and wider gaps between training and validation errors.

Linear Regression, while easy to interpret and respond fast, underestimates the data significantly.

Based on the metrics analysis, **Random Forest** can handle both general trends and day-to-day variability very well so does it achieve **the best balance between accuracy, stability, and flexibility**, making it the most reliable model for our predictive task.

Speaking to general drawbacks, all 4 models rely solely on historical data and don't account for real-world shocks like holidays, public events, or sudden weather changes. Also, the dataset only reflects city-level rental totals, which limits spatial insight into station-level behavior or regional patterns.

Next Step:

1. Move from daily to hourly prediction by using the hour.csv dataset. This would help the model capture short-term fluctuations in bike demand more effectively, such as rush hour peaks or midday drops.
2. Integrate external data sources/API, including weather forecasts, public event schedules, or unexpected disruptions (e.g., holidays or road closures). These inputs can improve the model's adaptability to real-world changes that aren't visible in historical patterns alone.
3. Deploy the model in an interactive environment, such as an online-version dashboard with auto-updating function. This would allow users to explore predictions dynamically, adjust inputs

in real time, and use the model as a practical decision-making tool for fleet allocation or demand forecasting.

Dataset Reference:

"Bike Sharing." UCI Machine Learning Repository, archive.ics.uci.edu/dataset/275/bike+sharing+dataset. Accessed 1 May 2025.