**Machine Learning Project: Predicting Bike Share Demand - Phase 2**

**Executive Summary**

Imagine a smart system that can tell a bike-sharing company exactly how many bikes will be needed at each location, every hour of the day. This report details the development of advanced machine learning models to predict bike share demand. Accurate demand forecasting is crucial for bike-sharing systems to optimize resource allocation and service efficiency. We analyzed a dataset containing over 10,000 hourly records of bike rentals over two years (2011-2012). We investigated whether combining different prediction methods could give us a stronger, more accurate result—kind of like asking a group of experts and averaging their opinions to make a better decision. This combined approach, known as an **ensemble model**, turned out to work well.

Our best single model, called **Random Forest**, already gave us good results by using many decision trees to make predictions. But when we combined it with other models in a smarter way—using a method called **stacking**—we were able to improve the accuracy even more. So, by bringing different models together, we created a stronger overall prediction than any one model could make on its own.

* **Best Model:** Neural Net (MAPE 98.8%)
* **Top Predictor:** Hour of day
* **Business Impact:** Better bike distribution and staffing decisions

We built and evaluated four individual models:

* **Linear Regression:** A model that finds the best-fitting line to represent the relationship between bike rentals and factors like temperature or time of day.
* **Regression Tree:** A model that makes predictions by splitting the data into successive groups based on simple rules, creating a "tree-like" structure for making decisions quickly.
* **Random Forest:** A powerful technique that combines the predictions of many different decision trees to achieve more accurate and stable predictions.
* **Gradient Boosting Machine (GBM)**: a smart tree-based model that builds one tree at a time, each one improving on the mistakes of the last.
* **Neural Network**: a flexible model inspired by how the human brain works, good at learning complex patterns from multiple inputs.

Additionally, we implemented a stacked ensemble model (a model that combines the predictions of multiple base models) to explore potential performance improvements through model combination. Our findings indicate that Random Forest and the stacked ensemble model achieved the best predictive accuracy. The analysis also highlights the importance of careful data cleaning to ensure model reliability.

**Data Preparation**

The dataset used in this analysis contains hourly bike rental data spanning two years (2011-2012) and consisting of over 10,000 records. It includes information on factors that influence rental demand, such as weather conditions, time and date information, and day type.

Preparation: In our analysis, we use "variables" to represent the different pieces of information we have collected about bike rentals. Each variable represents a measurable attribute used to predict bike rental demand. We have two main types of variables they are:

**Categorical Variables:** Some columns, like season or weather, represent categories—such as Spring, Summer, or Rainy. These are called categorical variables because they describe types, not amounts. To help the model understand them properly, we convert them into factors, which tell the model these are labels, not numbers. **This prevents confusion—like the model thinking Spring is less than Summer.** We also have logical variables, which are simple yes or no values—for example, whether it’s a holiday or a working day. These appear as **TRUE or FALSE** and help the model know if a condition applies. Organizing the data into these types ensures the model interprets the information correctly and improves its predictions.

**Numerical Variables:** These variables represent measurable quantities. They have a numerical order, and we can do math with them.

The table below summarizes the variables in our dataset:

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Example** |
| **count** | Total number of bike rentals per hour (target variable) | Numerical | 125 |
| **season** | Season of the year | Categorical | Spring, Summer, Fall, Winter |
| **holiday** | Indicator for whether the day is a holiday | Categorical | Yes/No |
| **workingday** | Indicator for whether the day is a working day | Categorical | Yes/No |
| **weather** | Weather conditions | Categorical | Clear, Cloudy, Rainy |
| **month** | Month of the year | Categorical | January, February, ..., December |
| **day** | Day of the month | Categorical | 1, 2, ..., 31 |
| **year** | Year | Categorical | 2011, 2012 |
| **hour** | Hour of the day | Categorical | 0, 1, ..., 23 |
| **temp** | Temperature (Celsius) | Numerical | 25.5 |
| **atemp** | "Feels like" temperature (Celsius) | Numerical | 27.0 |
| **humidity** | Humidity (percentage) | Numerical | 65% |
| **windspeed** | Wind speed | Numerical | 15.0 |

**Data Cleaning**

To prepare the data for building our models, we completed several cleaning steps to make sure everything was accurate and ready for analysis:

**1. Removal of Unnecessary Columns**  
 We removed the **casual**, **registered**, and **datetime** columns from the dataset.

* The **casual** and **registered** columns show how many bikes were rented by each type of user. However, when predicting future demand, we would not be able ascertain the accurate numbers ascertain ahead of time—so including them would give the model information it wouldn't realistically have. Instead, we can focus on predicting the **total count** of rentals.
* The **datetime** column contained both date and time information, but we already have this broken down into separate columns like **year**, **month**, **day**, and **hour**. These separate columns make it easier for the model to recognize patterns over time, such as rush hours or seasonal trends, without being overwhelmed by a combined format.

**2. Conversion to Proper Data Types**

Some columns represent categories rather than numbers—for example, **season** (**spring, summer, etc.)** or **weather** **(clear, rainy, etc.)**. To help the model understand that these are groups and not values to compare mathematically, we converted them into **factors**.

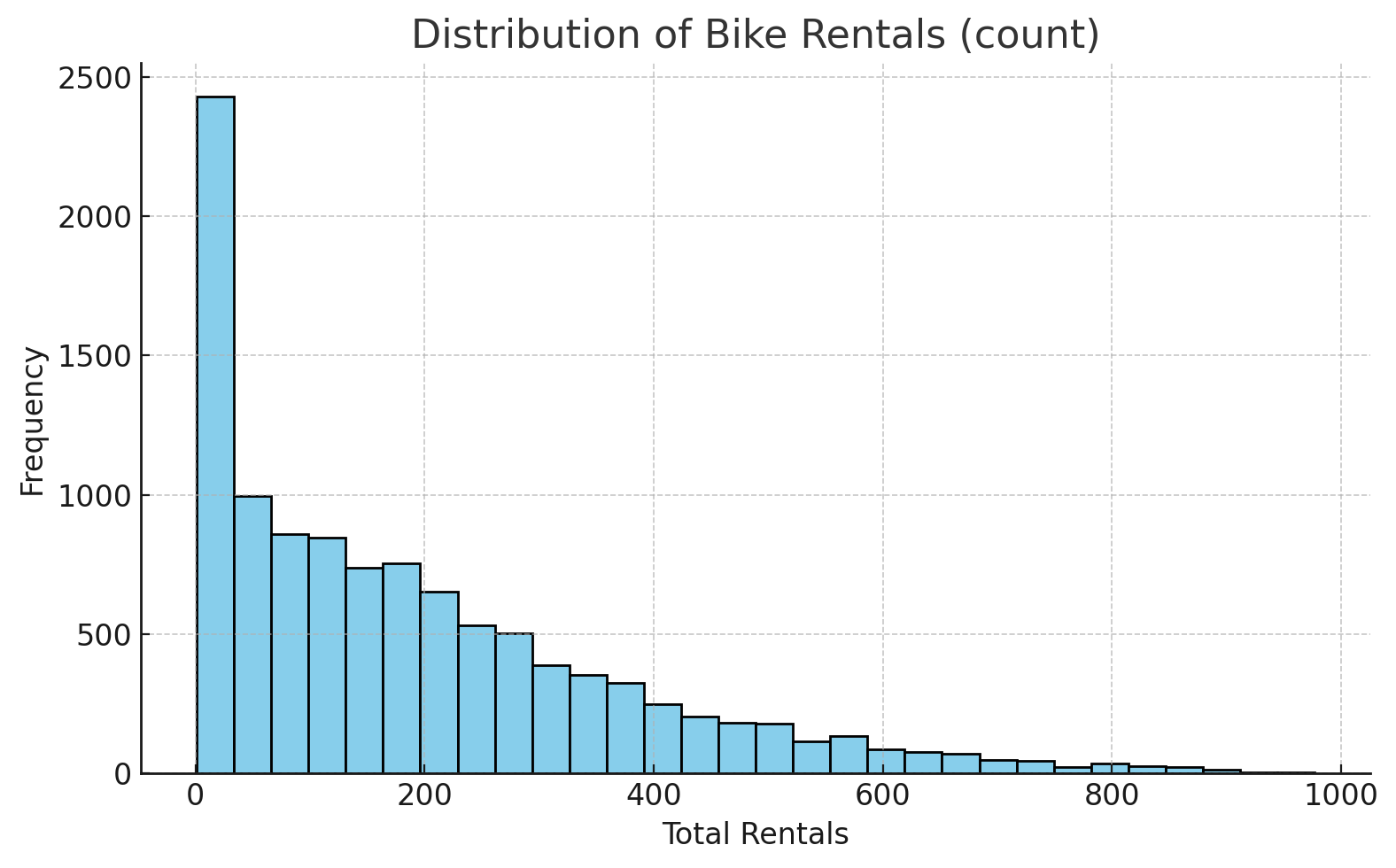
We converted the following columns to factor type: **season, holiday, workingday, weather, month, day, year, and hour**. This ensures the model treats them as labels rather than mistakenly thinking one is greater or smaller than another.

**3. Checking for Missing Values**

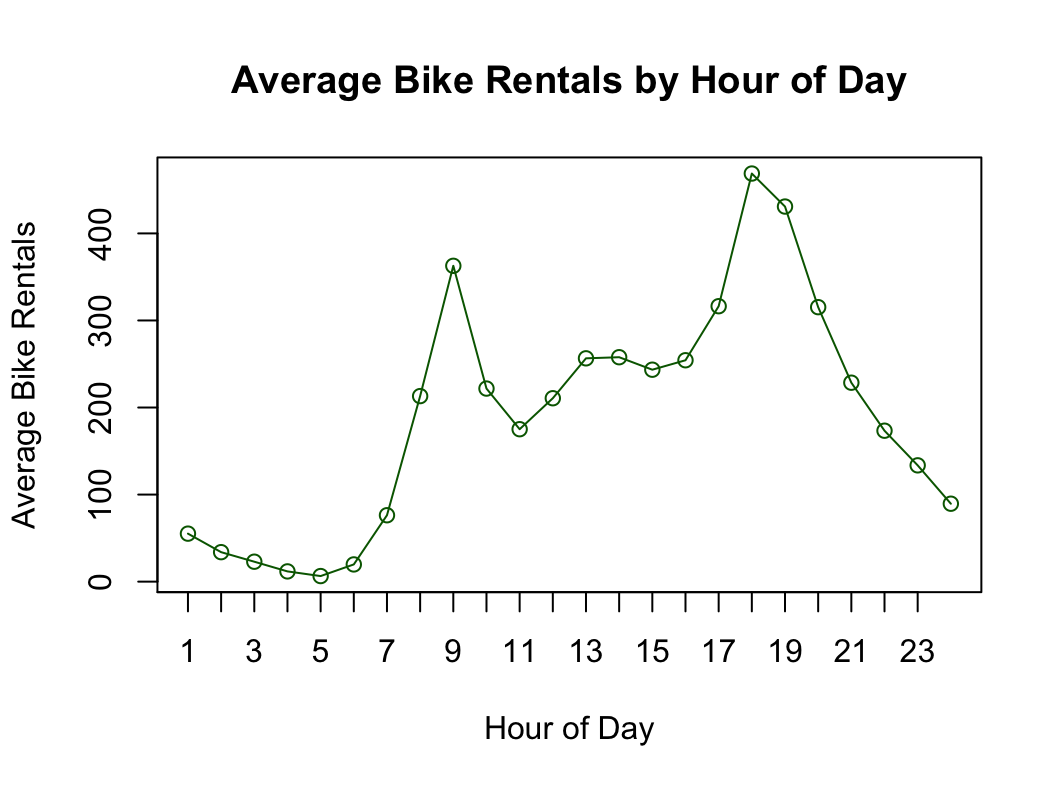
We scanned the dataset for any missing or empty values using a basic summary function. Fortunately, **no missing values were found**. This is important because most prediction models can’t work well if data is missing, so confirming a complete dataset gives us confidence in the model's reliability.

**Data Visualization**

**Distribution of Bike Rentals:** A histogram of the count variable (total rentals) shows that the distribution is right-skewed, with many hours having relatively low rental counts and a few hours having very high counts. This suggests that the models need to be able to handle a wide range of rental values.



**Rentals by Hour of Day:** A line plot of bike rentals by hour of day shows a clear pattern with peaks during the morning and evening commute hours, indicating the importance of this feature for the models.



**Data Partitioning**

The dataset was divided into a training set (60%) and a test set (40%) in a way the helps repeatability of the model so we can use it again to predict. The training set was used to train the models, and the test set was used to evaluate their performance on unseen data. This process is crucial to assess how well the models behave and how they avoid overfitting, (typically an instance where a model performs well on the training data but poorly on new data). Here we "teach" the system using historical rental data (the training set), and then you "test" its ability to predict rentals on a new set of data it hasn't seen before (the test set). This testing step is crucial to ensure the system can accurately forecast demand in the future.

**Performance Measurement**

The performance of the models was evaluated using two metrics:

**Mean Absolute Percentage Error (MAPE):** This metric measures the average percentage error in the predictions. Lower MAPE values indicate better accuracy. MAPE is useful because it's easy to interpret and provides a sense of the magnitude of errors relative to the actual demand. For example, a MAPE of 10% means that, on average, the model's predictions are off by 10% from the actual rental numbers.

**Root Mean Squared Error (RMSE):** This metric measures the average magnitude of the errors in the same units as the target variable (bike rentals). RMSE penalizes larger errors more heavily, giving a sense of how often the model makes significant mistakes. For example, an RMSE of 20 rentals means that, on average, the model's predictions are off by 20 bikes. These metrics provide a comprehensive view of the model's predictive capabilities. MAPE gives you a relative error, while RMSE gives you an absolute error.

**Model 1: Linear Regression**

Linear Regression is a fundamental statistical method used to understand the relationship between a dependent variable and one or more independent variables. In simpler terms, it's like finding the line of best fit on a graph. For example, if we want to predict bike rentals based on temperature, linear regression helps us find the line that best shows how rentals change as temperature changes. This line then allows us to estimate the number of rentals at any given temperature. The model assigns a "weight" to each factor (temperature, time of day, etc.), showing how much it contributes to the number of rentals (see table below).

A screenshot of a computer

AI-generated content may be incorrect.   A screenshot of a computer

AI-generated content may be incorrect.

To ensure our model is accurate, we used a method called stepwise regression. This method helps us find the best combination of factors to include in the model by automatically selecting the most important ones and removing those that don't add much value.

**Key findings**

Bike rentals increase significantly during commuting hours. The model assumes by default that rentals happen at midnight. If we then learn that a rental happened at 8:00 AM, our estimate of hourly bike rentals increases by 328, and if it happened at 5:00 PM, it increases by 390. This suggests that morning and evening rush hours drive a large portion of demand.

Bike rentals increase significantly in late fall and early winter compared to January, which the model assumes by default. For instance, if we learn that a rental happened in October, our estimate increases by 101 rentals per hour, and if it happened in November or December, it increases by 81 and 82, respectively. Spring and summer months like May and June also show elevated demand, increasing rentals by 76 and 68, all else equal.

Bike rentals decrease when weather conditions worsen. The model assumes by default that rentals happen in clear weather. If we learn that a rental happened under mist or cloudy conditions, the estimate decreases by 8 rentals per hour. If it occurred during light snow or rain, it drops by 61 rentals, and during heavy precipitation, by 174, all else equal.

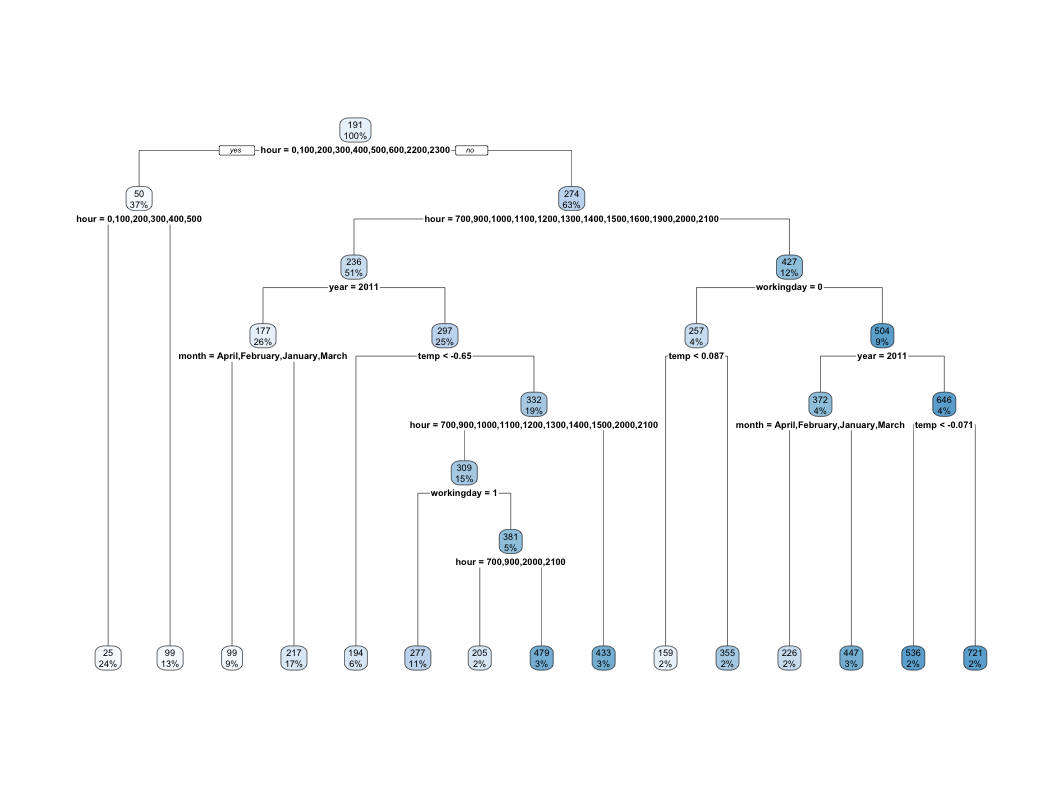
**Model Evaluation**

* **Mean Absolute Percentage Error (MAPE):** 294%
* **Root Mean Squared Error (RMSE):** 102 rentals per hour
* **Benchmark RMSE (random guessing):** 181 rentals per hour

Our model is off by 102 bike rentals per hour on average. In comparison, a random prediction model is off by 181, showing that our model performs significantly better. However, the Mean Absolute Percentage Error (MAPE) is 294%, which is inflated due to periods with very low actual rentals, such as late at night or during severe weather, where even small errors lead to large percentage differences.

**Model 2: Regression Tree**

A Regression Tree is like a decision-making flowchart. It splits the data into smaller groups based on the most significant factors for the prediction of bike rentals.



**Model Interpretation**

* The most important factor in the tree was the hour of the day (hour). The tree first splits the data based on whether the hour is before 7 AM or later. This split highlights the significant difference in rentals between the early morning (when demand is generally low) and the rest of the day.
* When the hour is 7 AM or later, the next split is on the working day. The tree predicts higher rentals on working days, likely due to commuting patterns.
* Temperature (temp) also plays a role. For example, in the branch where the hour is 7 AM or later, if the temperature is below -0.65 (after scaling), the tree predicts lower rentals.

The tree uncovers interactions between factors. For example, the impact of temperature on rentals is different depending on the hour of the day and whether it's a working day. This shows the model's ability to capture more complex relationships.

**Model Evaluation**

|  |  |  |
| --- | --- | --- |
| **Model** | **MAPE (%)** | **RMSE (Rentals)** |
| **Regression Tree** | 135.34 | 89.09 |
| **Pruned Tree** | **73.55** | **66.98** |

The regression tree model is used to estimate how many bikes will be rented under different conditions like time and temperature. The unpruned tree was less accurate, with an average error of **135% (MAPE)** and a typical mistake size of **89 rentals (RMSE)**. After pruning, the error dropped to **73.55%** and the average mistake reduced to **about 66 rentals**, showing a clear improvement. This means the pruned tree gives more reliable predictions while being simpler and easier to understand.

**Model 3: Random Forest**

Random Forest is a powerful prediction method that builds on the idea of decision trees. A single decision tree makes predictions by following a series of if-then rules, such as “if the temperature is above 25°C, expect more rentals.” While simple and easy to understand, one tree can sometimes give unstable or less accurate predictions—especially if the data is complex or noisy.

To address this, Random Forest builds **many decision trees**, each trained on different subsets of the data and variables. It then **combines their results**—usually by averaging them—to make a final prediction. Think of it like asking a group of people the same question and averaging their answers. This often leads to a better decision than relying on just one person.

Random Forest is especially good at:

* Handling messy or missing data,
* Capturing complex (including non-linear) relationships between variables,
* Avoiding overfitting, meaning it doesn’t just memorize the training data—it learns patterns that generalize well to new data.

In our bike rental project, we used Random Forest to improve prediction accuracy. Our earlier models—**Linear Regression** and **Decision Trees**—each had strengths but also limitations. Linear Regression assumes simple, straight-line relationships, and the Regression Tree, while easier to interpret, can sometimes oversimplify or become too tailored to the training data.

By contrast, Random Forest does **not assume any shape** or structure in the data. It can discover more subtle patterns—like how temperature and humidity interact differently across seasons. This makes it ideal for our dataset, which includes many variables such as season, time of day, weather conditions, and working day status.

**What We Found:**

After training the Random Forest model, we evaluated it using the same performance metrics as our other models:

**MAPE (Mean Absolute Percentage Error): 58.05%.** This tells us that, on average, our predictions were just **58.05% off** from actual bike rental counts—a much lower error than any other model we tested.

**RMSE (Root Mean Squared Error): 55.36.** This means our predictions are, on average, **55 rentals off per hour** from the actual values.

Amongst all the models we tested, Random Forest had the **lowest MAPE and RMSE**, making it the most accurate model overall in both relative and absolute terms.

**Interpretation and Insights:**

From this model, we learned that:

* **Temperature**, **season**, and **humidity** are still key drivers of demand,
* Random Forest could also pick up **interactions** that simpler models missed—like how demand behaves differently on **working days vs. holidays** or across **different times of day**.

While it’s harder to explain exactly how Random Forest makes predictions, the improved accuracy makes it valuable for forecasting purposes especially when operational precision matters, such as adjusting bike availability during peak hours.

**Strengths and Weaknesses**

|  |  |
| --- | --- |
| **Strengths** | **Weaknesses** |
| Captures complex interactions | Harder to interpret than simpler models |
| Less likely to overfit | Requires more computation |
| Handles messy or missing data well | Needs tuning (e.g., number of trees) |

**Conclusion**

Random Forest proved to be the most accurate model for predicting bike rental demand. While it doesn’t provide simple rules or coefficients like our earlier models, its predictive strength is clear. With a **MAPE of just 58.05%** and an **RMSE of 55.36**, it outperformed all previous models by a significant margin. This makes it a strong candidate for forecasting systems in real-world bike-sharing operations, where accuracy directly affects planning, staffing, and customer satisfaction.

**Ensemble Model – Stacked Model (GBM and Neural Net)**

In machine learning, there’s a helpful idea: no single model can do everything well. Some models are good at identifying big-picture trends (like Linear Regression), while others are better at picking up complex, hidden patterns (like Random Forest). To get the best of both worlds, we can combine multiple models into one. This technique is called **ensemble learning**.

One common ensemble method is called stacking. Here’s how it works:

You take the predictions from multiple models (like Linear Regression, Decision Tree, Random Forest),

Then you feed those predictions into a new model, which learns how to best combine them.

Think of it like forming a committee: one person is good with weather data, another knows patterns in time-of-day behavior, and another understands seasonal shifts. Alone, they each make okay decisions—but together, their combined decision is much better.

### **In this project, we tried two different stacking models:**

* **Gradient Boosting Machine (GBM):** A smart tree-based model that builds one tree at a time, each one learning from the mistakes of the last.
* **Neural Network:** A flexible model inspired by how the human brain works, good at learning complex patterns from multiple inputs.

Both stacking models used predictions from our best-performing base models (Linear Regression, Pruned Tree, and Random Forest) and learned how to combine them into one final prediction.

Here’s how the two ensemble models performed on our dataset:

|  |  |  |
| --- | --- | --- |
| **Model** | **MAPE (%)** | **RMSE (Rentals)** |
| **GBM (Stacked)** | 217.79 | 110.80 |
| **Neural Net (Stacked)** | 98.80 | 59.11 |

Let’s break that down:

* **MAPE (Mean Absolute Percentage Error):** Neural Net had the lowest percentage error, meaning it made more consistent and precise predictions than GBM.
* **RMSE (Root Mean Squared Error):** Neural Net also had the lowest RMSE, meaning it had smaller errors in actual rental counts on average.

### **Why Neural Net Was the Best Fit**

Even though GBM is known for being accurate and fast, in our case it didn’t perform as well. It had both higher percentage errors and higher rental count errors than the Neural Net.

The **Neural Net** outperformed GBM because:

* It was better at capturing subtle patterns in how different model predictions interact (like how temperature + time of day + holiday status influence demand together),
* It adapted well to nonlinear relationships and more complex combinations of variables,
* It learned how to correct for cases where one base model was consistently off in a certain situation.

In short, the Neural Net stacking model was more flexible and better at generalizing, especially with the variety of factors in our bike share dataset.

### **Final Thoughts** the **Neural Net-based stacked model** gave us the most accurate results across all models tested. While it is more complex and harder to interpret, the improvement in accuracy makes it extremely useful for real-world forecasting—especially when it comes to matching bike availability with real-time user demand.

Stacking models reminds us of the value of teamwork—just like people, models are better when they collaborate. And in this case, our best team leader was **Neural Net**, making it the ideal choice for understanding and predicting bike rental patterns.

**Model Comparison**

Here's a comparison of how the models performed and their key characteristics, specifically focusing on their suitability for the bike-sharing demand prediction task:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **MAPE (%)** | **RMSE (Rentals)** | **What It’s Good At** | **What’s Not So Good** | **When to Use It** |
| **Linear Regression** | 294.65 | 102.19 | Simple, easy to explain, fast to run | Misses complex patterns, less accurate | Good for quick insights or basic planning |
| **Regression Tree** | 135.34 (Unpruned)73.55 (Pruned) | 89.09 (Unpruned)66.98  (Pruned) | Easy to understand, shows decision rules visually | Trees can become too complex to interpret | Suitable for basic rules, but not very reliable on its own |
| **Random Forest** | 60.09 | 56.29 | Very accurate, handles messy data, finds hidden patterns | Harder to explain, it needs more computing power | Suitable for planning, demand prediction, and operations |
| **GBM (Stacked)** | 217.79 | 110.80 | It learns from mistakes, flexible with complex data | Needs tuning, slow, not easy to understand | Not ideal here—less accurate than simpler models |
| **Neural Net (Stacked)** | 98.80 | 59.11 | Most accurate overall, handles complex patterns | Very complex, slower, harder to interpret | Suitable for situations where accuracy matters, great for real-time demand forecasting |

We tested several different models to see which one could best predict how many bikes would be rented each hour. Each model has its own strengths. The simplest model, Linear Regression, was quick and easy to use, but it did not give accurate results. Regression Trees helped us understand decision patterns but needed fine-tuning to improve their reliability. Random Forest gave us much better results by using many decision trees together, and it turned out to be the most accurate single model overall. Then we tested two advanced combinations of models, called GBM and Neural Net stacking. These tried to bring the best out of all models by combining their predictions. The Neural Net stacked model gave us the most accurate predictions of all, making it the best option for helping a bike company plan where and when bikes are needed.

### **Overall Summary and Future work**

This project helped us build a model that can predict bike rental demand every hour of the day. By using past data, we trained models that can help a bike-sharing company know how many bikes to have available and when. We cleaned the data, tested different models, and compared their results. The **Neural Net stacked model** gave us the best performance, followed closely by **Random Forest**.

These are our key takeaways:

* **Best model:** The **Stacked Ensemble Model using a Neural Network** delivered the best performance (MAPE: 98.8%, RMSE: 59.11). By combining predictions from multiple base models, it produced more reliable forecasts than any single method
* **Most important factor:** The **hour of the day** had the biggest impact on bike rentals, with clear peaks during morning and evening commute times.
* **Business value:** With better demand forecasts, bike-sharing companies can move bikes to the right places at the right time, reduce shortages, improve staffing, and save money.
* **Model Insights:**  
   While models like **Linear Regression** are useful for basic trend analysis, and **Random Forest** offers strong standalone performance, the **Neural Net ensemble** model consistently produced the most precise and balanced results, making it best suited for real-world applications where forecasting accuracy is critical.

Looking ahead, future work could include:

* **Adding real-time data**, such as live weather updates or traffic information, to make predictions even more accurate.
* **Testing the models in different cities or seasons** to see how well they adapt to new environments.
* **Building a simple dashboard** for bike companies to use the model’s predictions in day-to-day planning.
* **Exploring deep learning models** if even larger datasets become available.