Capstone Project Bike Sharing

Demand Prediction using regression

**Shubham Joshi**

**Introduction:**

Currently rental bikes are introduced in many urban cities for the enhancement of mobility comfort. The purpose of this movement is to modernize cities and encourage people to head to a green world. The goal is to facilitate commuting in the city and reduce the number of cars and the pollution.

It is important to make the rental bike available and accessible to the public, as it provides many alternatives to commuters in metropolises. There are a lot of advantages to bike rents, it is convenient because it permits people not to keep the bike all day long, whether it is at work or at school. Furthermore, it is the healthiest way to travel and it has environmental benefits.

The target is the number of bikes rented per hour and date information. The dataset presents the company’s data between December the 1st of 2017 and finishes one year later. This study could have many aims for the company that could be seeing the results of the past year. It could also help them ameliorate themselves to become better and have a full satisfaction from customers.

# **1. Objective**

### The main objective of this project is to build an optimal predictive model is providing the city with a stable supply of rental bikes. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

In the Bike sharing demand dataset, the following attributes/features are available for us to begin with:

* Date: year-month-day

### Rented Bike count - Count of bikes rented at each hour

### Hour - Hour of the day

### Temperature-Temperature in Celsius

### Humidity - %

### Windspeed - m/s

### Visibility - 10m

### Dew point temperature - Celsius

### Solar radiation - MJ/m2

### Rainfall - mm

### Snowfall - cm

### Seasons - Winter, Spring, Summer, Autumn

### Holiday - Holiday/No holiday

### Functional Day – NoFunc (Non-Functional Hours), Fun (Functional hours)

**2. Exploratory Data Analysis:**

In terms of EDA, there is quite a bit of analysis and derivations that we have discovered and worked with. In the interest of keeping this document precise and on-the-dot, we shall cover the most significant logical findings of EDA. A more detailed approach of the EDA is available in the associated Python Notebook. Kindly refer to that, for more clarity.

**EDA Insights:**

**Working Day:**

for a Working Day where the rental count high at peak hours (Most demand for bike is in between 7 to 9 AM and 5 to 8 PM.)

**Non-working day:**

where rental count is more or less uniform across the day with a peak at around noon.

**Hour of the day:**

Bike rental count is mostly correlated with the time of the day. As indicated above, the count reaches a high point during peak hours on a working day and is mostly uniform during the day on a non-working day

**Season:**

We see highest number bike rentals in Fall (July to September) and Summer (April to June) Seasons and the lowest in Spring (January to March) season

**Weather:**

As one would expect, we see highest number of bike rentals on a clear day and the lowest on a snowy or rainy day

**Humidity**:

With increasing humidity, we see decrease in the number of bike rental count.

# **3. Feature Engineering**

All machine learning algorithms use some input data to create outputs. This input data comprises of various features, which are usually in the form of structured columns. Algorithms require features with some specific characteristics to work properly.

Here, the need for feature engineering arises. Feature engineering mainly have two goals:

* Preparing the proper input dataset, compatible with the machine learning algorithm requirements.
* Improving the performance of machine learning models.

We'll try adding and removing some features in this section in order to make a perfect data matrix we can pass to a machine learning model. We will try to interpret categorical features as numeric, so as to be passed to the ML models.

From the year column, we can easily extract the day, month and year of the record. As this data is limited to only one year so the year of the record is not that important.

From the year, we will extract day, month for our analysis.

We will divide our day column to two categories as weekday (all other days except Saturday and Sunday) and weekend (Saturday and Sunday), so that it will help us in reducing the features for **one hot encoding.**

After this we will observe the datatypes of our present features and change them according to our necessity.

* **Multicollinearity:**

We calculated **Variance Inflation Factor** to check multicollinearity. There was much multicollinearity in between temperature and dew point temperature features, so we had to drop Dew point Temperature feature from the dataset.

* **Transformation:**

As our Target variable ‘Rented bike count’ was moderately right skewed, so we tried some transformations and after some trails we found out that **square root** transformation was making the distribution somewhat similar to normal.

* **One hot Encoding:**

A one hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

We used one hot encoding for our all-categorical Features.

After doing some feature engineering we divided our dataset into two categories for making us easy to do EDA.

**1.Numerical Features:** 'Rented Bike Count', 'Temperature(°C)’, ‘Humidity (%)’, ‘Wind speed (m/s)’, ‘Visibility (10m)’, ‘Solar Radiation (MJ/m2)’, ‘Rainfall(mm)', 'Snowfall (cm)'.

**2.Categorical Features:** 'Hour’, ‘Seasons', 'Holiday’, ‘Functioning Day’, ‘Month', 'WeekdayorWeekend'

# **4.Model Selection:**

Now it’s time to implement the Machine Learning models and check the accuracy of each model to point out the best one out of all. In this project we have tried and tested 7 machine learning algorithms to predict the target variable, post which we then apply optimization techniques on the one that gives best r2 score out of all.

Following algorithms have been used for predictions: -

## **Linear Regression:**

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It’s used to predict values within a continuous range, rather than trying to classify them into categories. There are two main types:

Simple regression:

Simple linear regression uses traditional slope-intercept form, where m and b are the variables, our algorithm will try to “learn” to produce the most accurate predictions. x represents our input data and y represents our prediction.

Y = m \*x +b

Multivariable regression:

A more complex, multi-variable linear equation might look like this, where w represents the coefficients, or weights, our model will try to learn.

f (x, y, z) =w1x+w2y+w3z

* **Lasso Regression:**

Lasso regression stands for Least

Absolute Shrinkage and Selection Operator. It adds penalty term to the cost function. This term is the absolute sum of the coefficients. As the value of coefficients increases from 0 this term penalizes, cause model, to decrease the value of coefficients in order to reduce loss. The difference between ridge and lasso regression is that it tends to make coefficients to absolute zero as compared to Ridge which never sets the value of coefficient to absolute zero.

* **Limitation of Lasso Regression:**

Lasso sometimes struggles with some types of data. If the number of predictors (p) is greater than the number of observations (n), Lasso will pick at most n predictors as nonzero, even if all predictors are relevant (or may be used in the test set). If there are two or more highly collinear variables then LASSO regression select one of them randomly which is not good for the

interpretation of data

## **Ridge Regression:**

In Ridge regression, we add a penalty term which is equal to the square of the coefficient. The L2 term is equal to the square of the magnitude of the coefficients. We also add a coefficient lambda to control that penalty term. In this case if lambda is zero then the equation is the basic OLS else if lambda > 0 then it will add a constraint to the coefficient. As we increase the value of lambda this constraint causes the value of the coefficient to tend towards zero. This leads to trade-off of higher bias (dependencies on certain coefficients tend to be 0 and on certain coefficients tend to be very large, making the model less flexible) for lower variance.

* **Limitation of Ridge Regression:**

Ridge regression decreases the complexity of a model but does not reduce the number of variables since it never leads to a coefficient been zero rather only minimizes it. Hence, this model is not good for feature reduction.

* **Elastic Net Regression:**

Sometimes, the lasso regression can cause a small bias in the model where the prediction is too dependent upon a particular variable. In these cases, elastic Net is proved to better it combines the regularization of both lasso and Ridge. The advantage of that it does

not easily eliminate the high collinearity coefficient.

## **Decision Tree:**

Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

Decision trees are upside down which means the root is at the top and then this root is split into various several nodes. Decision trees are nothing but a bunch of if-else statements in layman terms. It checks if the condition is true and if it is then it goes to the next node attached to that decision.

In a Decision Tree diagram, we have:

**Root Node**: The first split which decides the entire population or sample data should further get divided into two or more homogeneous sets. In our case, the Outlook node.

**Splitting:** It is a process of dividing a node into two or more sub-nodes.

**Decision Node:** This node decides whether/when a sub-node splits into further sub-nodes or not. Here we have, Outlook node, Humidity node, and Windy node.

**Leaf:** Terminal Node that predicts the outcome (categorical or continuous value). The coloured nodes, i.e., Yes and No nodes, are the leaves.

* **Random Forest Regression:**

Random Forest is a technique that uses ensemble learning, that combines many weak classifiers to provide solutions to complex problems.

As the name suggests random forest consists of many decision trees. Rather than depending on one tree it takes the prediction from each tree and based on the majority votes of predictions, predicts the final output.

Random forests use the bagging method. It creates a subset of the original dataset, and the final output is based on majority ranking and hence the problem of overfitting is taken care of.

## **XGB Regression:**

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e., how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg: linear, and that for binary classification is reg: logistics. Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensembles learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancel out and better one sums up to form final good predictions.

# **5.Model Optimization:**

From the Scores we have seen earlier Random Forest has given by far the accuracy in predicting the target variable.

We have implemented Grid search Cross Validation upon Random Forest, Lasso and ridge to fine-tune prediction results from our model. To achieve that we'll use Grid Search CV that will help us find best hyperparameters values for our models.

Grid Search uses different combinations of all the specified hyperparameters and their values and calculates the performance for each combination and selects the best value for the hyperparameters. This makes the process time-consuming and expensive based on the number of hyperparameters involved

**6. Conclusion:**

We were able to see that the linear algorithms were not performing better even with the hyperparameter tunning and cross validation.

Linear regression, lasso and ridge gave r2 scores 0.77, 0.78 and 0.78 respectively in the test data.

Out of the tree-based algorithms, the

Random Forest Regressor was providing an optimal solution towards achieving our Objective. We were able to achieve an R2 score of 0.99 in the train split, and 0.92 in the test split. We also noticed that even in the case of Decision tree, we were able to achieve an R2 score of 0.78 in the test split. XGB regressor gave an r2 score of 0.86 in the test data.

We then implemented Grid Search Cross Validation on the Random Forest Regressor, to further try to optimize the model, and were able to achieve an R2 score of 0.96 in the train split, and 0.90 in the test split.

Finally, we conclude Random Forest (with and without GridsearchCV) to be the best model to achieve our objective.

# **7. References:**

* Python Pandas Documentation https://pandas.pydata.org/pandas

-docs/stable

* Python Matplotlib Documentation

https://matplotlib.org/stable/index.

html

* Python SkLearn Documentation

https://scikit-learn.org/stable