**Bike Sharing Demand Prediction**

**Indugopal Maity & Md. Tousif Ali,**

**Data science trainees**

**Alma Better, Bangalore**

**Abstract:**

A bicycle-sharing system is a [shared transport](https://en.wikipedia.org/wiki/Shared_transport) service in which [bicycles](https://en.wikipedia.org/wiki/Bicycles) are made available for shared use to individuals on a short-term basis for a price or free. Many bike share systems allow people to borrow a bike from a "dock" and return it at another dock belonging to the same system. Docks are special [bike racks](https://en.wikipedia.org/wiki/Bicycle_parking_rack) that lock the bike, and only release it by computer control. The user enters payment information, and the computer unlocks a bike. The user returns the bike by placing it in the dock, which locks it in place. Other systems are dockless.

***Keywords: machine learning, bike sharing, temperature, seasons, holidays, functioning day.***

**1.Problem Statement**

Data, “SeaulBikeData.csv”, provided by an Alma Better. Customers/Users can download their respective app on smartphones and book a bike from anywhere nearest hub of that particular bike sharing platform and can return to the nearest hub of destination.

The main objective is to build a predictive model, which could help them in predicting the rush hour proactively. This would in turn help them in matching the availability of bikes with the right customers quickly and efficiently.

* Normal peak-hours
* How weathers are related (temperature, humidity, snowfall, rainfall and etc.)
* Events (Holiday or Non-holiday)
* Type of Day (Functioning and Non-Functioning Day)

**2. Introduction**

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 2000 bike-sharing programs around the world.

### Our goal here is to build a predictive model, which could help us in predicting the rush hour proactively.

# **3. How different time of a day work**

## **Demand for rides increases**

There are times when so many people are requesting bike ride on the road. High temperature, office hour, weather, humidity, and holiday. For instance, may cause unusually large numbers of people to want to request a ride with certain bike sharing company all at the same time (mainly in office hour).

**4. Steps involved:**

* **Exploratory Data Analysis**

After loading the dataset, we performed this method by comparing our target variable that is Rented Bike Count with other independent variables. This process helped us figuring out various aspects and relationships among the target and the independent variables. It gave us a better idea of which feature behaves in which manner compared to the target variable.

* **Null values Treatment**

Our dataset contains a large number of null values which might tend to disturb our accuracy hence we dropped them at the beginning of our project in order to get a better result.

* **Encoding of categorical columns**

We used One Hot Encoding to produce binary integers of 0 and 1 to encode our categorical features because categorical features that are in string format cannot be understood by the machine and needs to be converted to numerical format. Like, we have considered weekdays as “0” and weekends as “1”.

* **Feature Selection**

In these steps we used algorithms like “Random Forest” classifier to check the results of each feature i.e. which feature is more important compared to our model and which is of less importance.

* **Standardization of features**

Our main motive through this step was to scale our data into a uniform format that would allow us to utilize the data in a better way while performing fitting and applying different algorithms to it.

The basic goal was to enforce a level of consistency or uniformity to certain practices or operations within the selected environment.

* **Fitting different models**

For modelling we tried various classification algorithms like:

1. **Logistic Regression**
2. **LASSO Regression**
3. **RIDGE Regression**
4. **ELASTIC NET Regression**
5. **Decision Tree**
6. **Random Forest Classifier**
7. **XGBoost classifier**

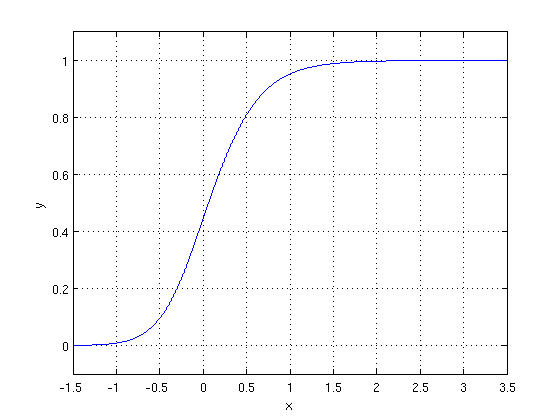
* **Tuning the hyperparameters for better accuracy**

Tuning the hyperparameters of respective algorithms is necessary for getting better accuracy and to avoid overfitting in case of tree based models, like Random Forest Classifier and XGBoost classifier.

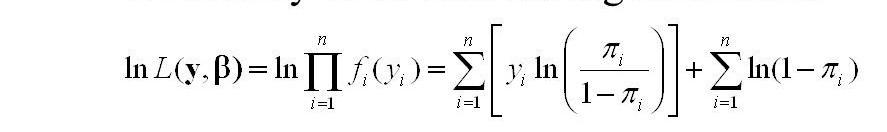
**5.1. Algorithms:**

1. **Logistic Regression:**

Logistic Regression is actually a classification algorithm that was given the name regression due to the fact that the mathematical formulation is very similar to linear regression. The function used in Logistic Regression is sigmoid function or the logistic function given by: f(x)= 1/1+e ^(-x)

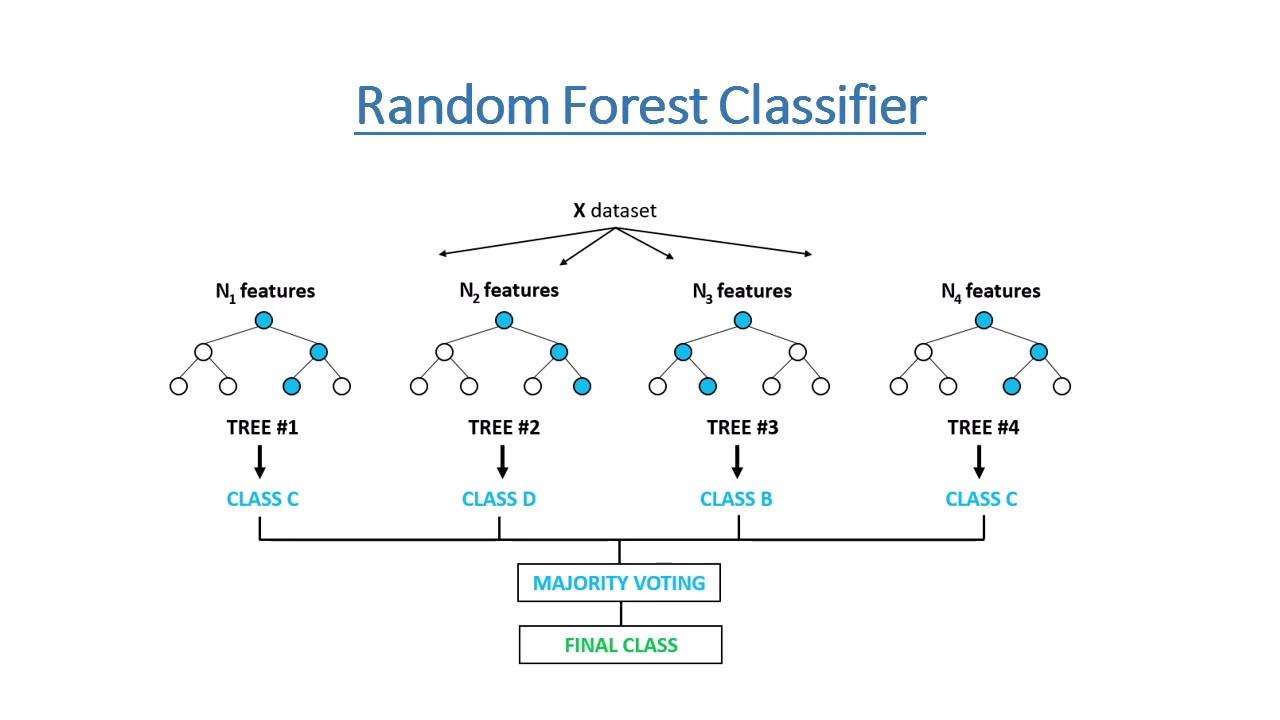


The optimization algorithm used is: Maximum Log Likelihood. We mostly take log likelihood in Logistic:



1. **Random Forest Classifier:**

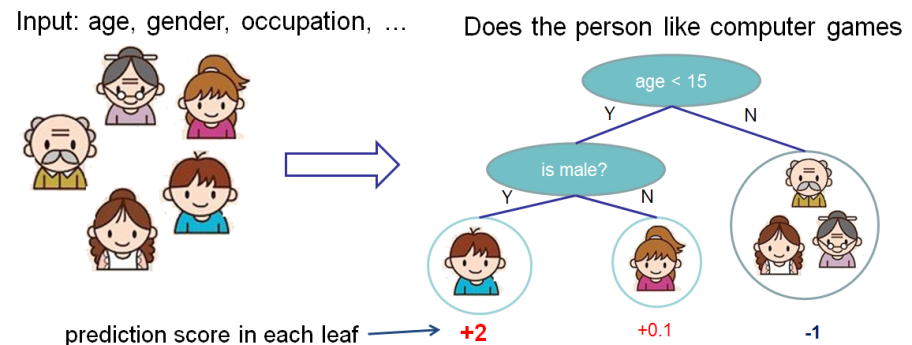
Random Forest is a bagging type of Decision Tree Algorithm that creates a number of decision trees from a randomly selected subset of the training set, collects the labels from these subsets and then averages the final prediction depending on the most number of times a label has been predicted out of all.



1. **XGBoost-**

To understand XGBoost we have to know gradient boosting beforehand.

**Gradient Boosting-** Gradient boosted trees consider the special case where the simple model is a decision tree.

****

In this case, there are going to be 2 kinds of parameters P: the weights at each leaf, w, and the number of leaves T in each tree (so that in the above example, T=3 and w=[2, 0.1, -1]). When building a decision tree, a challenge is to decide how to split a current leaf. For instance, in the above image, how could I add another layer to the (age > 15) leaf? A ‘greedy’ way to do this is to consider every possible split on the remaining features (so, gender and occupation), and calculate the new loss for each split; you could then pick the tree which most reduces your loss.

**XGBoost** is one of the fastest implementations of gradient boosting. trees. It does this by tackling one of the major inefficiencies of gradient boosted trees: considering the potential loss for all possible splits to create a new branch (especially if you consider the case where there are thousands of features, and therefore thousands of possible splits). XGBoost tackles this inefficiency by looking at the distribution of features across all data points in a leaf and using this information to reduce the search space of possible feature splits.

**5.2. Model performance:**

Model can be evaluated by various metrics such as:

**1.Confusion Matrix**- The confusion matrix is a table that summarizes how successful the classification modelis at predicting examples belonging to various classes. One axis of the confusion matrix is the label that the model predicted, and the other axis is the actual label.

**2. Precision/Recall**- Precision is the ratio of correct positive predictions to the overall number of positive predictions: TP/TP+FP

Recall is the ratio of correct positive predictions to the overall number of positive examples in the set: TP/FN+TP

1. **Accuracy**-

Accuracy is given by the number of correctly classified examples divided by the total number of classified examples. In terms of the confusion matrix, it is given by: TP+TN/TP+TN+FP+FN

**5.3. Hyper parameter tuning:**

Hyperparameters are sets of information that are used to control the way of learning an algorithm. Their definitions impact parameters of the models, seen as a way of learning, change from the new hyperparameters. This set of values affects performance, stability and interpretation of a model. Each algorithm requires a specific hyperparameters grid that can be adjusted according to the business problem. Hyperparameters alter the way a model learns to trigger this training algorithm after parameters to generate outputs.

We used Grid Search CV, Randomized Search CV and Bayesian Optimization for hyperparameter tuning. This also results in cross validation and in our case we divided the dataset into different folds. The best performance improvement among the three was by Bayesian Optimization.

1. **Grid Search CV-**Grid Search combines a selection of hyperparameters established by the scientist and runs through all of them to evaluate the model’s performance. Its advantage is that it is a simple technique that will go through all the programmed combinations. The biggest disadvantage is that it traverses a specific region of the parameter space and cannot understand which movement or which region of the space is important to optimize the model.
2. **Randomized Search CV-** In Random Search, the hyperparameters are chosen at random within a range of values that it can assume. The advantage of this method is that there is a greater chance of finding regions of the cost minimization space with more suitable hyperparameters, since the choice for each iteration is random. The disadvantage of this method is that the combination of hyperparameters is beyond the scientist’s control.
3. **Conclusion:**

That's it! We reached the end of our exercise.

Starting with loading the data so far we have done EDA, null values treatment, encoding of categorical columns, feature selection and then model building. In all of these models our accuracy revolves in the range of 95% to 98%.

And there is no such improvement in accuracy score even after hyperparameter tuning.

So the accuracy of our best model is 98%% which can be said to be good for this large. dataset. This performance could be due to various reasons like: no proper pattern of data, too much data, not enough relevant features.

**References-**

1. Google.