**Credit Card Default Prediction**

**Akash Salmuthe**

**Data Science Enthusiast**

**AlmaBetter, Bangalore**

**Abstract:**

In today’s world credit cards have become a lifeline to a lot of people so banks provide us with credit cards..

Our experiment can help understand what could be the reason for the classification of such labels by feature selection, data analysis and prediction with machine learning algorithms taking into account previous trends to determine the correct classification. For finding defaulters

***Keywords: machine learning ,Credit card default prediction, classified labels***

**1.Problem Statement**

Data provided by an Indian cab aggregator service Sigma Cabs. Their customers can download their app on smartphones and book a cab from anywhere in the cities they operate in. They, in turn, search for cabs from various service providers and provide the best option to their clients across available options. They have been in operation for a little less than a year now. During this period, they have captured surge pricing types from the service providers.

The main objective is to build a predictive model, which could help them in predicting the surge pricing type proactively. This would in turn help them in matching the right cabs with the right customers quickly and efficiently.

* Trip\_ID: ID for TRIP
* Trip\_Distance: The distance for the trip requested by the customer
* TypeofCab: Category of the cab requested by the customer
* CustomerSinceMonths: Customer using cab services since n months; 0 month means the current month
* LifeStyleIndex: Proprietary index created by Sigma Cabs showing the lifestyle of the customer based on their behaviour
* ConfidenceLifeStyle\_Index: Category showing confidence on the index mentioned above
* Destination\_Type: Sigma Cabs divides any destination into one of the 14 categories.
* Customer\_Rating: Average of lifetime ratings of the customer till date CancellationLast1Month: Number of trips cancelled by the customer in last 1 month
* Var1, Var2 and Var3: Continuous variables masked by the company. Can be used for modelling purposes
* Gender: Gender of the customer
* SurgePricingType: Target (can be of 3 types) - DV

**2. Introduction**

### The cab platforms adjust their prices using a specific algorithm which is real time and dynamic known as **“Surge Pricing”** or **“Dynamic Pricing”**. This algorithm automatically raises the price of a trip when the demand increases more than the supply.

### The surge algorithm generally outputs a multiplier which is adjusted along with the base fare, the price per mile and the price per minute to generate the final price. This price is communicated to the riders and the ride is initiated when they confirm the price shown. This surge multiplier is kept discrete and may range from 1.2 to the maximum allowed by the government based on geography.

### Our goal here is to build a predictive model, which could help Sigma Cabs in predicting the surge pricing type proactively.

## **3. Types of Pricing**

* Static Pricing
* Dynamic Pricing(Surge Pricing)

### The distance and travel time based taxi pricing scheme (Static Pricing) has been prevalent for decades. One major drawback of the current taxi price is that it fails to take the time of day into consideration while the demand in the market is time sensitive. So there is a need for Dynamic pricing.

## **4. Reasons for surge pricing**

The reasons for surge pricing are:

* normal peak-hours
* bad weather conditions (rain, snow, etc)
* events (concerts, movie-premiere)
* traffic conditions
* unseen emergencies and so on.

# **5. How Surge pricing works**

## **Demand for rides increases**

There are times when so many people are requesting rides that there aren’t enough cars on the road to help take them all. Bad weather, rush hour, and special events, for instance, may cause unusually large numbers of people to want to request a ride with Sigma all at the same time.

## **Prices go up**

In these cases of very high demand, prices may increase to help ensure that those who need a ride can get one. This system is called surge pricing, and it lets the app continue to be a reliable choice.

## **Riders pay more or wait**

Whenever rates are raised due to surge pricing, the app lets riders know. Some riders will choose to pay, while some will choose to wait a few minutes to see if the rates go back down.

**6. Steps involved:**

* **Exploratory Data Analysis**

After loading the dataset we performed this method by comparing our target variable that is Surge\_Pricing\_Type 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 inorder 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.

* **Feature Selection**

In these steps we used algorithms like ExtraTree 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.

Next we used Chi2 for categorical features and ANOVA for numerical features to select the best feature which we will be using further in our model.

* **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. **SVM Classifier**
3. **Random Forest Classifier**
4. **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.

* **SHAP Values for features**

We have applied SHAP value plots on the Random Forest model to determine the features that were most important while model building and the features that didn’t put much weight on the performance of our model.

**7.1. Algorithms:**

1. **Logistic Regression:**

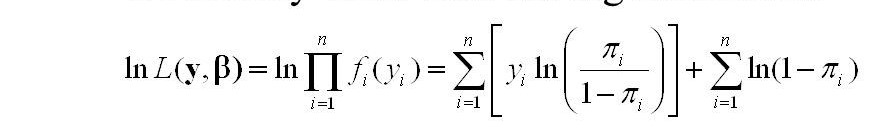
Logistic Regression is actually a classification algorithm that was given the name regression due to the fact that the mathematical formulation is very similar to linear regression.

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:

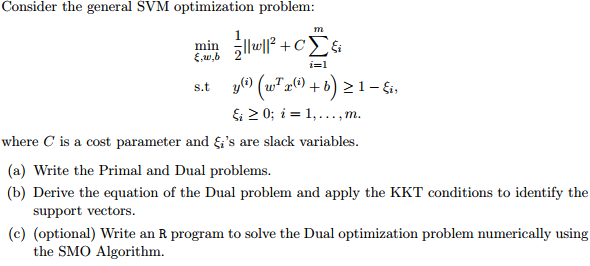


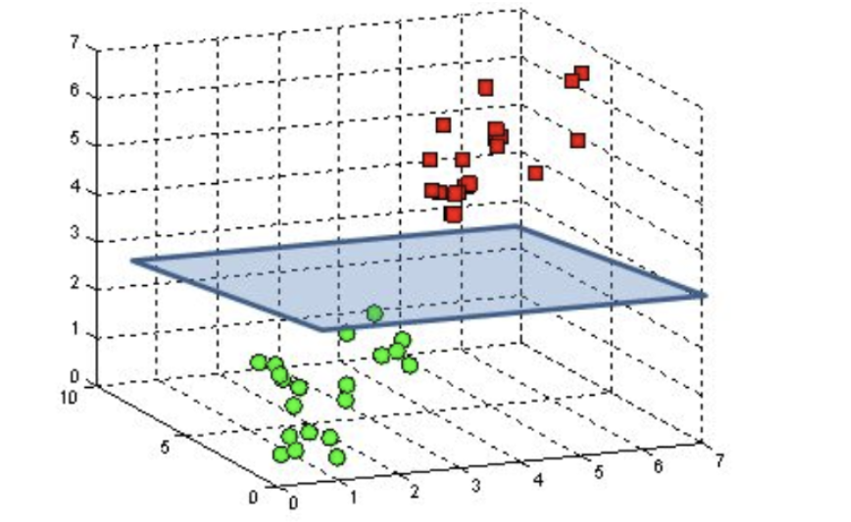
.

1. **Support Vector Machine Classifier:**

SVM is used mostly when the data cannot be linearly separated by logistic regression and the data has noise. This can be done by separating the data with a hyperplane at a higher order dimension.

In SVM we use the optimization algorithm as:





We use hinge loss to deal with the noise when the data isn’t linearly separable.

Kernel functions can be used to map data to higher dimensions when there is inherent non linearity.

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.

**7.2. 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.

1. **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

1. **Area under ROC Curve(AUC)**-

ROC curves use a combination of the true positive rate (the proportion of positive examples predicted correctly, defined exactly as recall) and false positive rate (the proportion of negative examples predicted incorrectly) to build up a summary picture of the classification performance.

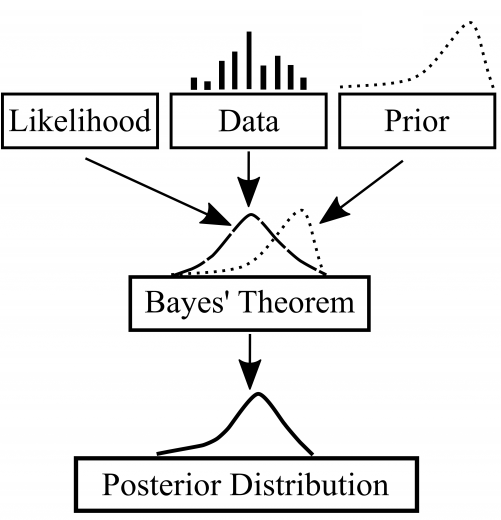
**7.3. Hyper parameter tuning:**

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

# **Bayesian Optimization-** Bayesian Hyperparameter optimization is a very efficient and interesting way to find good hyperparameters. In this approach, in naive interpretation way is to use a support model to find the best hyperparameters.A hyperparameter optimization process based on a probabilistic model, often Gaussian Process, will be used to find data from data observed in the later distribution of the performance of the given models or set of tested hyperparameters.



As it is a Bayesian process at each iteration, the distribution of the model’s performance in relation to the hyperparameters used is evaluated and a new probability distribution is generated. With this distribution it is possible to make a more appropriate choice of the set of values that we will use so that our algorithm learns in the best possible way.

**8. 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 70 to 74%.

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

So the accuracy of our best model is 73% 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. MachineLearningMastery
2. GeeksforGeeks
3. Analytics Vidhya