# Report A1

Roll Number : 2022102039

Name : Meet Gera

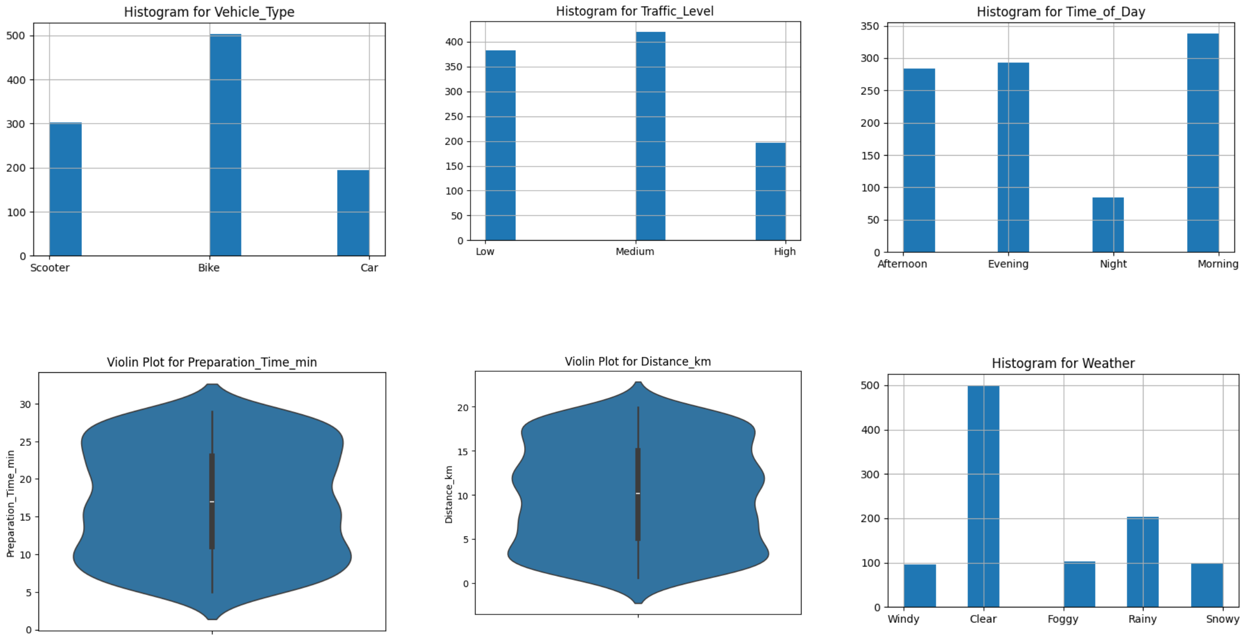
Course : SMAI

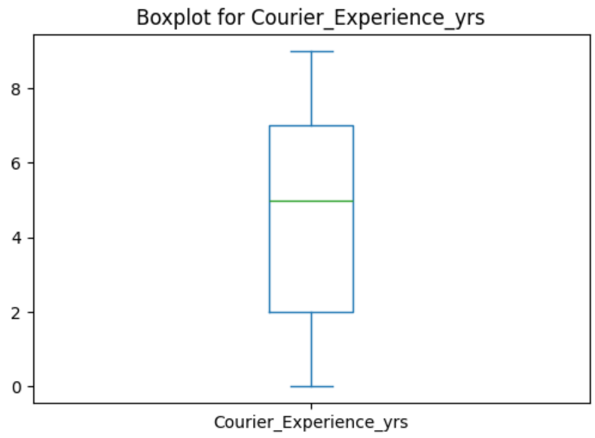
Assignment 1

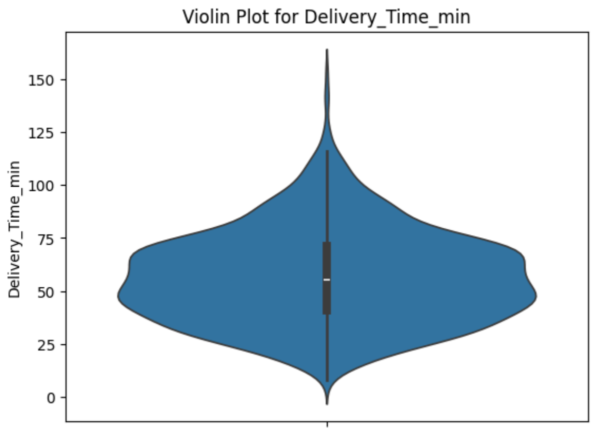
# Question 2

## 2.1 : EDA

### Feature Plots :







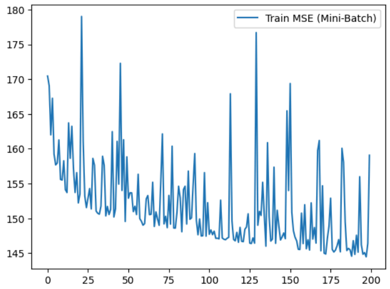
### Observation And Insights :

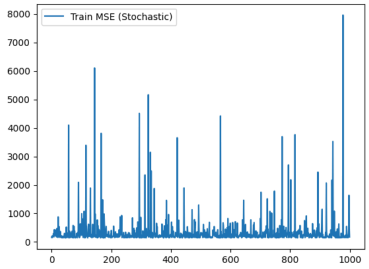
- Most deliveries happen during daytime , and that too best at clear weather conditions
- Two wheelers like bike are used more than 4 wheelers due to traffic and efficiency for food delivery
- Delivery time has shape similar to normal or Beta distribution

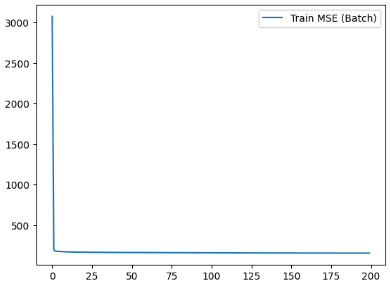
## 2.2 Linear Regression with Gradient Descent

### Final Test Set MSE and R-2 Score for each Method

| Method | MSE | R-2 | Extra info |
|---|---|---|---|
| Batch | 153.84 | 0.64 | lr = 0.01 , iter = 200 |
| Mini Batch | 150 | 0.65 | size = 32 |
| Stochastic | 149 | 0.65 | lr = 0.01 , iter = 1000 |

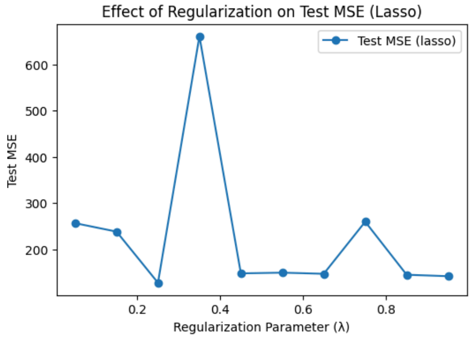


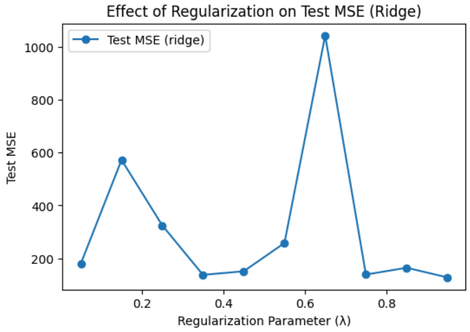




This is for Batch GD

## 2.3 Regularisation





The optimal values for both are visible from the MSE vs reg_param plot.

| Reg Param | MSE | R-2 | Extra info | Type |
|---|---|---|---|---|
| 0.45 | 150 | 0.65 | lr = 0.01 , iter = 200 | Ridge |
| 0.25 | 128 | 0.70 | lr = 0.01, iter = 200 | Lasso |

**Low Regularization (small λ)**: The model may overfit, capturing noise in the training data, leading to low training error but high test error.

**High Regularization (large λ)**: The model may underfit, oversimplifying patterns and leading to higher bias, which increases both training and test errors.

Here the initialisations chosen were random so differences can occur.

## 2.4 Report

1. Types of Gradient Descent Algorithms their advantages and disadvantages

1️⃣***Batch Gradient Descent (BGD)***

*Batch Gradient Descent computes the gradient using the entire training dataset before updating model parameters. This makes it computationally expensive for large datasets but provides stable convergence. However, it can be slow and get stuck in local minima, especially in non-convex loss functions.*

2️⃣***Stochastic Gradient Descent (SGD)***

*SGD updates the model parameters after computing the gradient for each individual data point. This makes it faster and more suitable for large datasets but introduces high variance in updates, leading to a noisy optimization path. The randomness can help escape local minima but may prevent reaching the optimal solution precisely.*

3️⃣***Mini-Batch Gradient Descent (MBGD)***

*MBGD balances the trade-offs by computing gradients on small batches of data, offering a middle ground between BGD's stability and SGD's efficiency. It speeds up convergence while reducing noise in updates, making it widely used in deep learning. However, choosing an appropriate batch size is crucial to ensure efficient training without excessive memory usage.*

2. Gradient Descent that converges the fastest

*Stochastic Gradient Descent (SGD) converges the fastest per update since it updates parameters after each data point, but it may be noisy and less stable compared to Mini-Batch Gradient Descent (MBGD) and Batch Gradient* *descent.*

*For this exercise the noise proved to be a big factor at bringing unstability here as seen in the plots. In this exercise practice Batch converged the fastest.*

3. How did Lasso and Ridge regularisation influence the model? What is the optimal lambda (amongst the ones you have chosen) for Lasso and Ridge based on test performance?

*Lasso and Ridge regularisation influenced the model by reducing overfitting and improving generalisation. Lasso (L1) shrank some coefficients to zero, promoting sparsity, while Ridge (L2) reduced coefficients without setting them to zero, leading to a more stable model.*

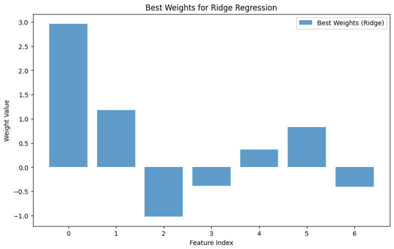
*The optimal lambda for Lasso and Ridge is the one that minimizes test MSE while maintaining a good balance between bias and variance. Based on test performance is 0.25 for lasso and 0.45 for Ridge.*

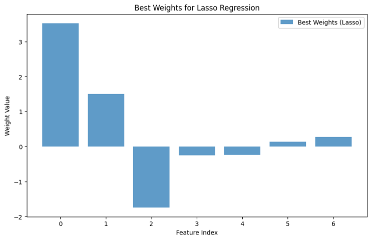
4. How does scaling of features affect model performance?

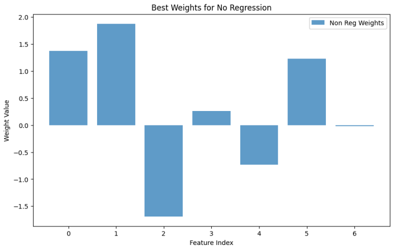
*Scaling features improves model performance by ensuring that all variables contribute equally to the learning process. It prevents large-magnitude features from dominating smaller ones, leading to faster convergence in gradient-based optimizations and improving numerical stability.*

*In models like linear regression, scaling does not affect the final predictions but speeds up training. However, for regularised models (like Ridge and Lasso) and distance-based models (like k-NN and SVM), proper scaling is crucial to avoid bias toward features with larger magnitudes.*

5. Using a Barplot, describe how the trained model weights (1 per feature) vary in case of best performing model for Lasso and Ridge, compared to non-regularized model (best).







6. Based on values of feature weights, analyze the effect of each feature on delivery time. Are there any features that have 0 (or almost 0) effect?

Feature Index 6 have low weight for all Ridge, Lasso and No Regression. They might have very less effect. Courier Experience in years has very low effect on Delivery Time.

Distance_km has the highest net weight among 3 and it makes sense that delivery time is highly related to distance and weather.

# Question 3

## 3.1.1 Classification:

### KNN {test_set, train_set}

| K | Distance Metric | Accuracy |
|---|---|---|
| 1 | Euclidean | 90.48% |
| 5 | Euclidean | 91.82% |
| 10 | Euclidean | 91.94% |
| 1 | Cosine | 90.48% |
| 5 | Cosine | 91.82% |
| 10 | Cosine | 91.94% |

## 3.1.2 Retrieval

### Text to Image Retrieval

MRR is 1.0000

Precision is 0.9740

Hit Rate is 1.0000

### KNN {test_set, text_set}

| K | Metric | Accuracy |
|---|---|---|
| 1 | Euclidean | 87.81 |
| 1 | Cosine | 87.81 |

### Image to Image Retrieval

MRR is 0.9348

Precision is 0.8411

Hit Rate is 0.9996

Average Comparisons per Query: 50000

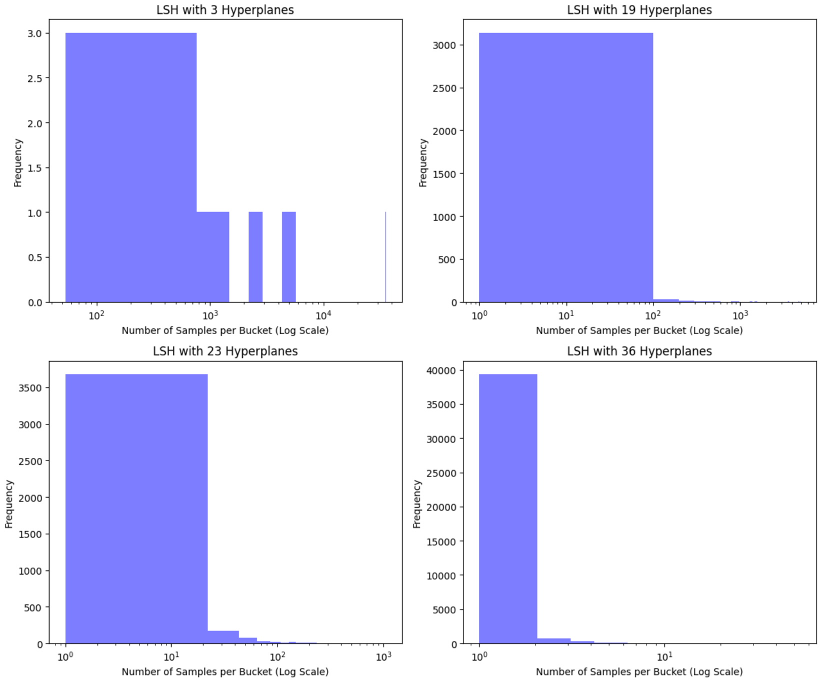
## 3.2 LSH

MRR is 0.6013

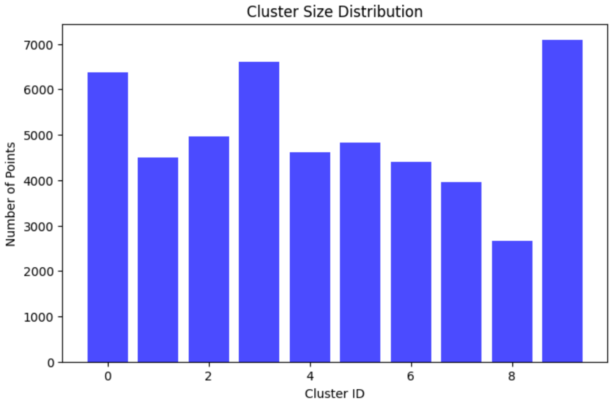
Precision is 0.3023

Hit Rate is 0.9950

Average Comparisons per Query: 225.9646



## 3.3 IVF

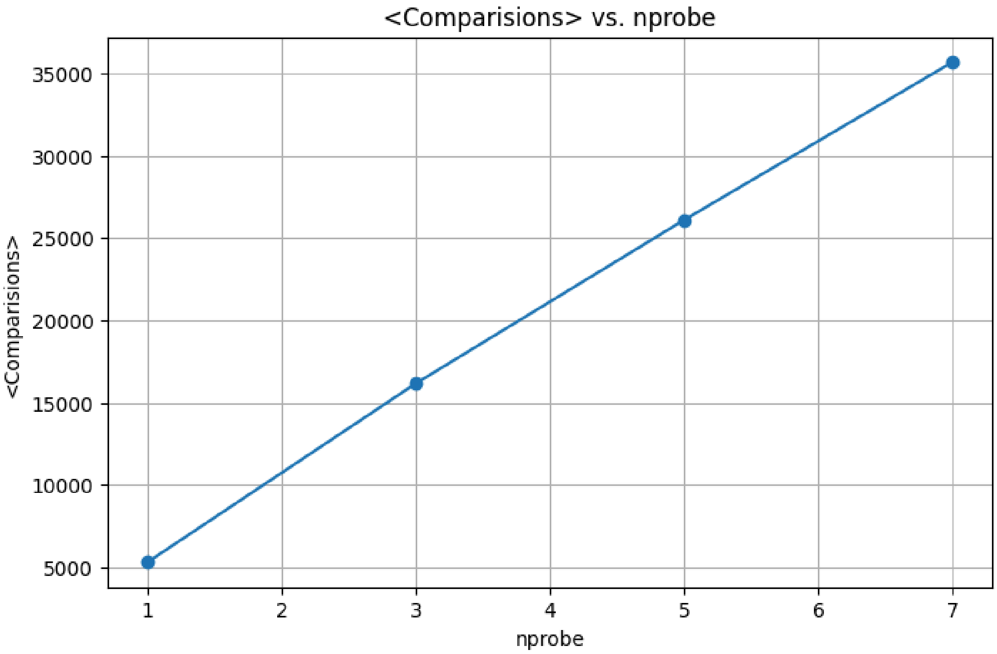


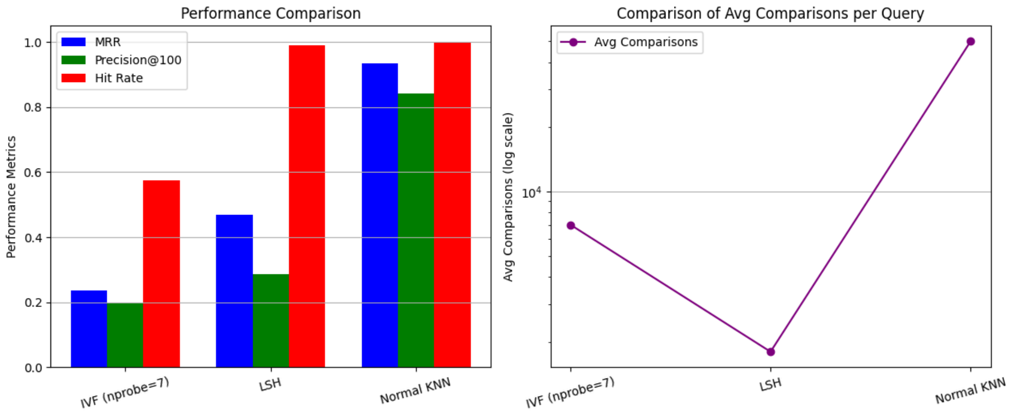
Average comparisons  for nprobe = 3  is 3232.40

Average comparisons for nprobe = 7 is 7132.08

## 3.4 Analysis

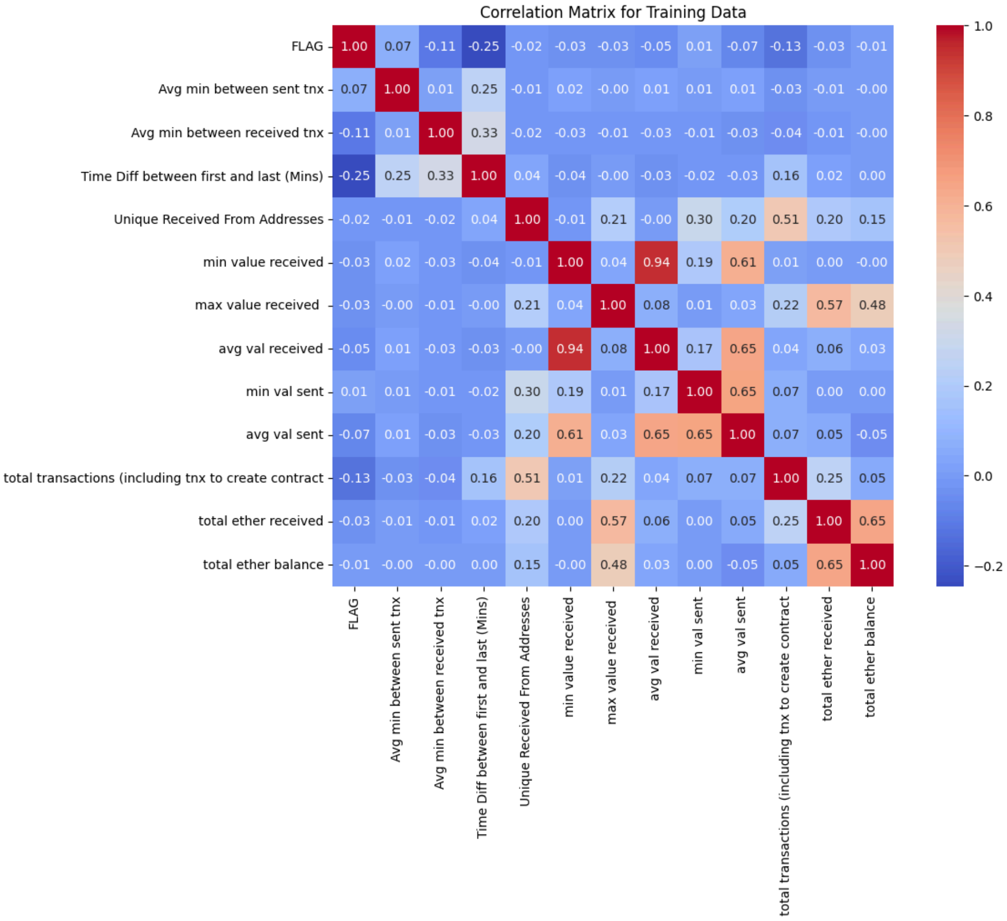
| Method | MRR | Precision | HitRate | Avg Comparisons |
|---|---|---|---|---|
| LSH | 0.6013 | 0.3023 | 0.9950 | 225.96 |
| KNN | 0.9348 | 0.8411 | 0.9996 | 50k |
| IVF(nprobe = 7) | 0.237 | 0.199 | 0.576 | 7k |



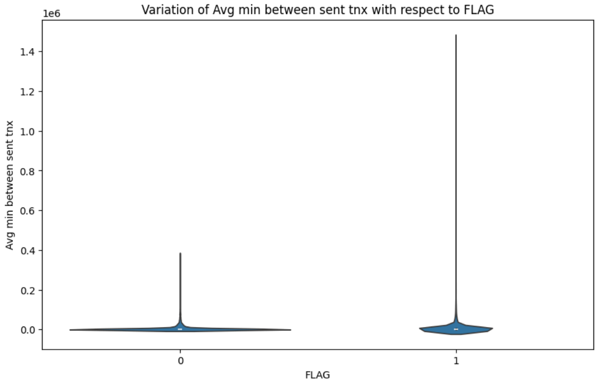


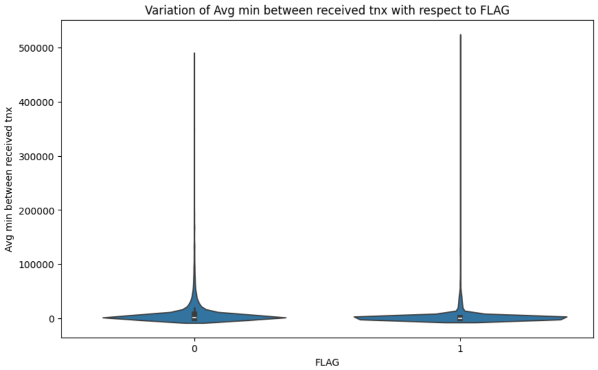
# Question 4

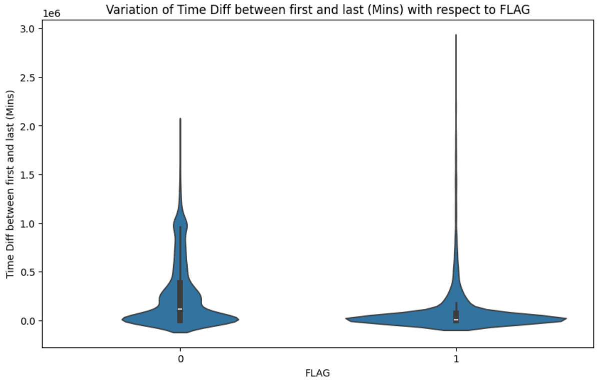
## 4.1 Preprocessing and EDA

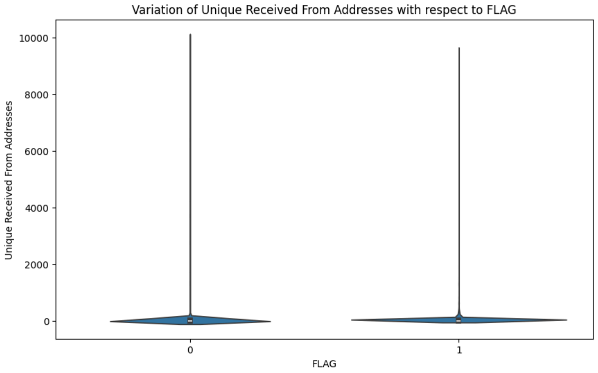


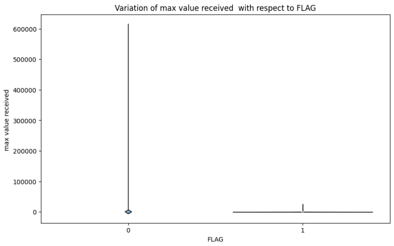
- FLAG has low correlation with other features highest one being with Time difference between first and last transaction in mins
- Among features themselves there is high correlation between avg value received and min value received.

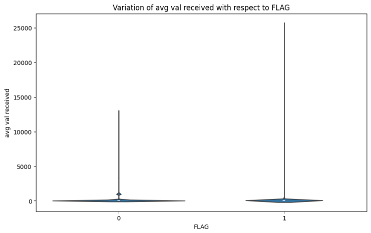


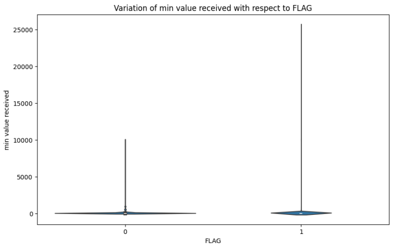


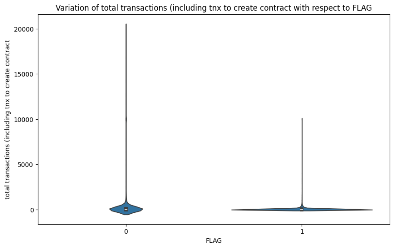


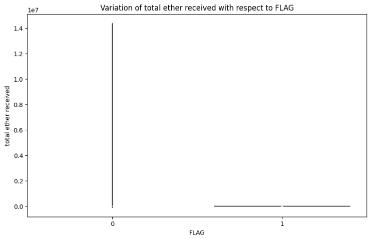


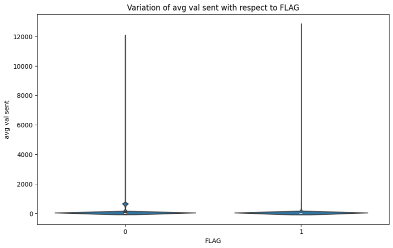












## 4.2 Fraud Detection Model

Comparison with Scikit-learn's built in implementation

| **Criterion** | **Model** | **Train Accuracy** | **Validation Accuracy** | **Test Accuracy** | **Time Taken (s)** |
|---|---|---|---|---|---|
| **Entropy** | Scratch | 0.8829 | 0.8861 | 0.8933 | 0.2613 |
| | Scikit-learn | 0.8915 | 0.8808 | 0.8940 | 0.0800 |
| **Gini** | Scratch | 0.8829 | 0.8861 | 0.8933 | 0.2398 |
| | Scikit-learn | 0.9090 | 0.8839 | 0.9025 | 0.0636 |

Mostly the trees made by entropy and gini's criteria for scratch and scikit-learn's implementation are same, inbuilt implementation is slightly faster and slightly better at accuracy possibly due to better search for splitting thresholds as I tried to go through all possible threshold and selecting best, both were giving exactly same results.

## Hyperparameter Tuning

### Accuracy for Different Hyperparameters

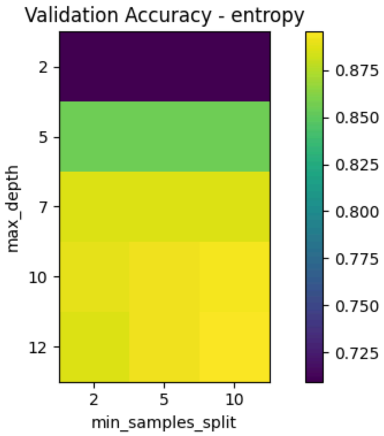
**Criterion: Entropy**

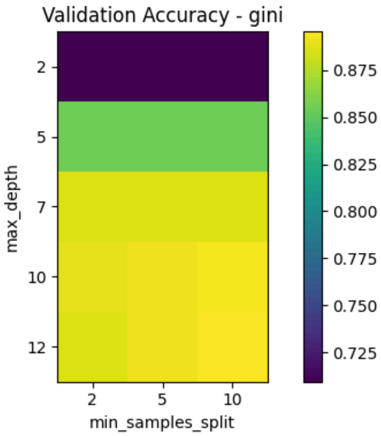
| **max_depth** | **min_samples_split = 2** | **min_samples_split = 5** | **min_samples_split = 10** |
|---|---|---|---|
| 2 | 0.7095 | 0.7095 | 0.7095 |
| 5 | 0.8552 | 0.8552 | 0.8552 |
| 7 | 0.8861 | 0.8861 | 0.8861 |
| 10 | 0.8892 | 0.8918 | 0.8927 |
| 12 | 0.8865 | 0.8918 | 0.8954 |

**Criterion: Gini**

| **max_depth** | **min_samples_split = 2** | **min_samples_split = 5** | **min_samples_split = 10** |
|---|---|---|---|
| 2 | 0.7095 | 0.7095 | 0.7095 |
| 5 | 0.8552 | 0.8552 | 0.8552 |
| 7 | 0.8861 | 0.8861 | 0.8861 |
| 10 | 0.8892 | 0.8918 | 0.8927 |
| 12 | 0.8865 | 0.8918 | 0.8954 |

This table makes it easier to compare results across different hyperparameter settings.

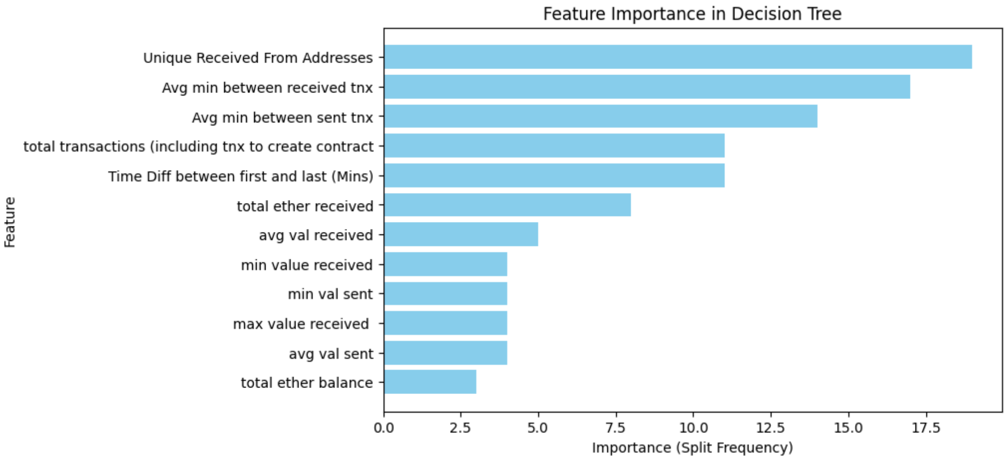


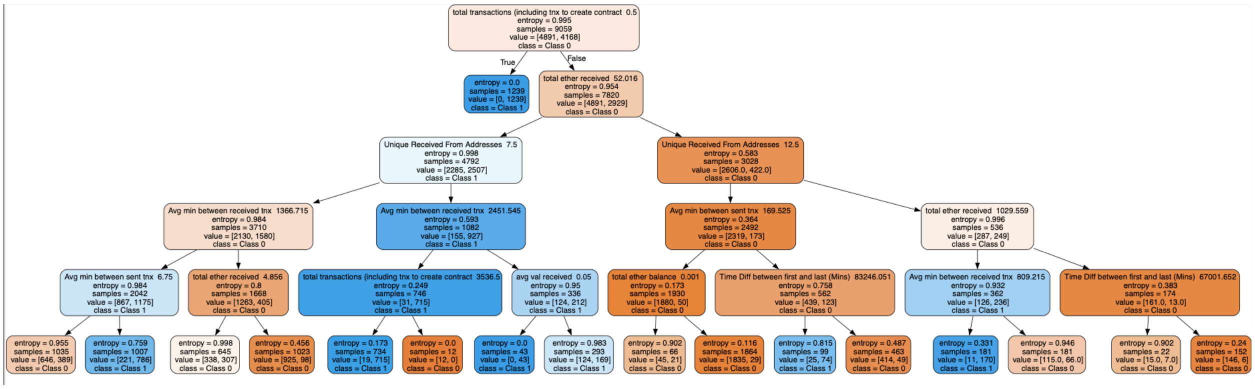


This heat map describes the accuracy of tree wrt different max_depth and min_sample_split, as we can see above 7 the max_depth does not offer any advantage but takes longer time and can lead to overfitting and decrease in accuracy for any type of Distance metric, both Euclidean and gini.

## Explanation for Result

For Low max_depth the accuracy is low is information gain by less number of questions answered is less than needed to correctly deduce answer. As depth is increased the information gain increases but this information gain saturates after certain point of time, and for a diverse test set would lead to overfitting errors.




total transactions (including tnx to create contract 0.5 entropy = 0.995 samples = 9059 value = [4891, 4168] class = Class 0
True
False
entropy = 0.0 samples = 1239 value = [0, 1239] class = Class 1
total ether received 52.016 entropy = 0.954 samples = 7820 value = [4891, 2929] class = Class 0
Unique Received From Addresses 7.5 entropy = 0.998 samples = 4792 value = [2285, 2507] class = Class 1
Unique Received From Addresses 12.5 entropy = 0.583 samples = 3028 value = [2606.0, 422.0] class = Class 0
Avg min between received tnx 1366.715 entropy = 0.984 samples = 3710 value = [2130, 1580] class = Class 0
Avg min between received tnx 2451.545 entropy = 0.593 samples = 1082 value = [155, 927] class = Class 1
Avg min between sent tnx 169.525 entropy = 0.364 samples = 2492 value = [2319, 173] class = Class 0
total ether received 1029.559 entropy = 0.996 samples = 536 value = [287, 249] class = Class 0
Avg min between sent tnx 6.75 entropy = 0.984 samples = 2042 value = [867, 1175] class = Class 1
total ether received 4.856 entropy = 0.8 samples = 1668 value = [1263, 405] class = Class 0
total transactions (including tnx to create contract 3536.5 entropy = 0.249 samples = 746 value = [31, 715] class = Class 1
avg val received 0.05 entropy = 0.95 samples = 336 value = [124, 212] class = Class 1
total ether balance 0.001 entropy = 0.173 samples = 1930 value = [1880, 50] class = Class 0
Time Diff between first and last (Mins) 83246.051 entropy = 0.758 samples = 562 value = [439, 123] class = Class 0
Avg min between received tnx 809.215 entropy = 0.932 samples = 362 value = [126, 236] class = Class 1
Time Diff between first and last (Mins) 67001.652 entropy = 0.383 samples = 174 value = [161.0, 13.0] class = Class 0
entropy = 0.955 samples = 1035 value = [646, 389] class = Class 0
entropy = 0.759 samples = 1007 value = [221, 786] class = Class 1
entropy = 0.998 samples = 645 value = [338, 307] class = Class 0
entropy = 0.456 samples = 1023 value = [925, 98] class = Class 0
entropy = 0.173 samples = 734 value = [19, 715] class = Class 1
entropy = 0.0 samples = 12 value = [12, 0] class = Class 0
entropy = 0.0 samples = 43 value = [0, 43] class = Class 1
entropy = 0.983 samples = 293 value = [124, 169] class = Class 1
entropy = 0.902 samples = 66 value = [45, 21] class = Class 0
entropy = 0.116 samples = 1864 value = [1835, 29] class = Class 0
entropy = 0.815 samples = 99 value = [25, 74] class = Class 1
entropy = 0.487 samples = 463 value = [414, 49] class = Class 0
entropy = 0.331 samples = 181 value = [11, 170] class = Class 1
entropy = 0.946 samples = 181 value = [115.0, 66.0] class = Class 0
entropy = 0.902 samples = 22 value = [15.0, 7.0] class = Class 0
entropy = 0.24 samples = 152 value = [146, 6] class = Class 0

# Question 5

## Impact of #Clusters and Compactness Factor

The **number of clusters** in SLIC (Simple Linear Iterative Clustering) directly affects the granularity of superpixels. A higher number of clusters results in smaller, more localized superpixels that better capture fine details, while a lower number produces larger, coarser superpixels that generalize regions more. Choosing the right number depends on the application—too few clusters may lose important details, whereas too many may increase computational cost and noise.

The **compactness factor** controls the balance between color similarity and spatial proximity when forming superpixels. A lower compactness favors color similarity, leading to irregularly shaped clusters that adhere closely to object boundaries. In contrast, a higher compactness forces superpixels to be more spatially regular, resembling a grid-like structure. Adjusting this parameter is crucial—too low may cause over-segmentation near edges, while too high may result in boundary leakage.

## RGB vs LAB

In the RGB space, color differences are not perceived uniformly by the human eye, leading to distortions in distance measurement. Unlike the Lab space, where Euclidean distance aligns with perceptual similarity, RGB distances may misrepresent color variations—some color changes appear more significant than others. When using RGB for clustering (e.g., in SLIC superpixel segmentation), boundaries may become less accurate, with superpixels forming irregularly due to the non-uniform sensitivity of RGB components. This occurs because RGB channels mix luminance and chromatic information non-linearly, whereas Lab separates them, making Lab a more reliable choice for perceptual color differences.

## Average number of iterations used per frame change

The new video is also much smoother.

Normal took 100s for 11 frames ~ 9.09 s / per frame

Optimized took 30s for 10 further frames ~ 3s per frame

Thus it took 3x less time thus approximately 3x Less number of iterations per frame in optimized version.

Link For all Video and frame Resources : https://iiithydstudents-my.sharepoint.com/:f:/g/personal/meet_gera_students_iiit_ac_in/EiSs4EHgHCtJtVMXdGCSR9MBegpzj3I_oT3nRNw-lJyFfg?e=PPVFgg