IE7275 - Project Report

1. Motivation:

In today's digital era, E-commerce giants strive to convert millions of browsing users into purchasers. Therefore, it's crucial for these companies to determine the value of each user interaction to maximize success. This will even help companies to optimize their offerings, and tailor product recommendations to better serve the needs of their customers.

2.Goal:

The goal is to analyze patterns within the clickstream data and predict the potential purchasing amount or price for each user.

The goal is to analyze patterns within the clickstream data and label each user interaction based on the potential purchase they might make in that ongoing session.

Phase 1: Dataset Selection and Preprocessing

3. Data Source:

Dataset Title: Clickstream data for online shopping.

The dataset is taken from the UCI dataset repository: Link to the dataset

4. Data Description:

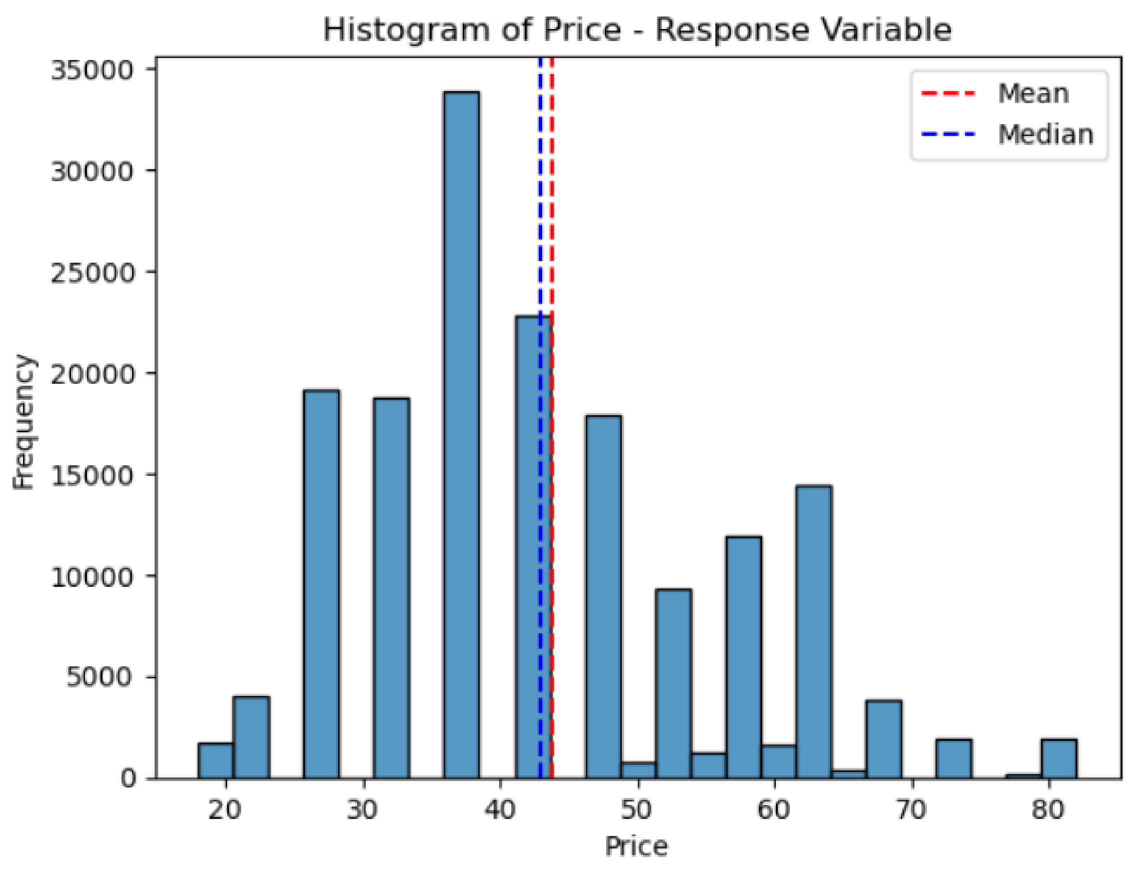
The clickstream data is collected from an online clothing store for pregnant women. It is collected by tracking each user's click on a digital product like a web application / mobile application. The data is from five months of 2008 and has **165,474** rows and **14** attributes. The **original response variable** is the **price** attribute, which is a **continuous variable**. However, we are interested in a discrete response variable that represents the potential purchase amount in that user session. Hence, we map the values of the continuous variable to discrete states and create a new variable called the **price category**. Therefore, it is a **Classification Problem.**

The following table summarizes the data attributes from the dataset.

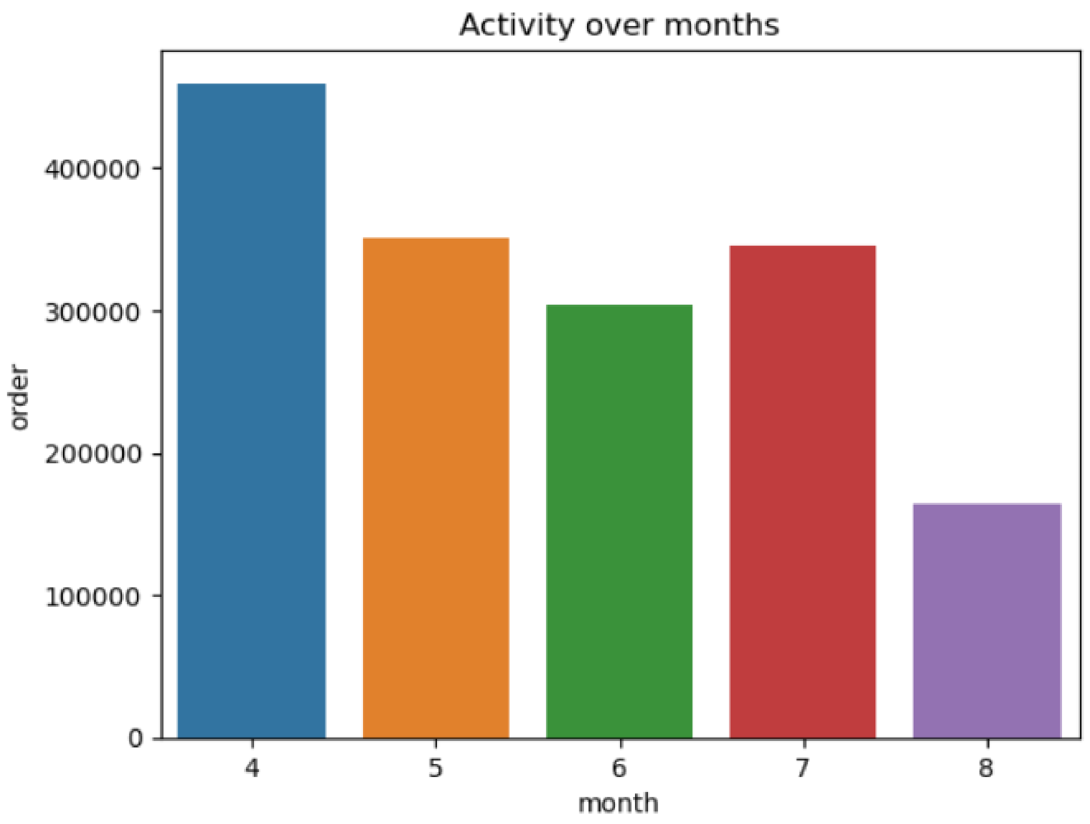
Attribute	Description	Data Type	Range of Values
Year	Year when the data was collected.	Integer/Numeric	Just one - 2008
Month	Month when the data was collected.	Integer/Numeric	April (4) to August (8)

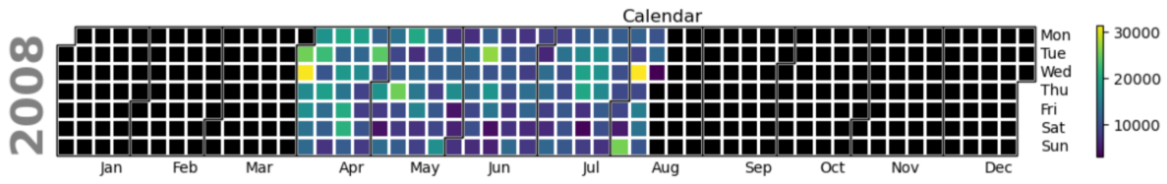
Day	Day when the data was collected.	Integer/Numeric	1 to 31
Order	Number of clicks during each session.	Integer/Numeric	1 to 195
Country	Country of origin for an IP Address.	Integer/Numeric	1 to 47
Session ID	Unique ID of each web session.	Integer/Numeric	1 to 24026
Page 1 (Main category)	Main product category.	Integer/Numeric	1 to 4: 1-trousers 2-skirts 3-blouses 4-sale
Page 2 (Clothing model)	Product code for each product under main category.	String	Alphanumeric: ` B4`. 217 Unique strings
Color	Color of each product.	Integer/Numeric	1 to 14: 1 – beige; 2 - black 3 – blue; 4 – brown 5 – burgundy; 6 - gray 7 – green; 8 - navy blue; 9 - of many colors; 10 - olive 11 – pink; 12 - red 13 – violet;14 - white
Location	Location of the picture w.r.t six frames in which the page is divided.	Integer/Numeric	1 to 6: 1 - top left 2 - top in the middle 3 - top right 4 - bottom left 5 - bottom in the middle 6 - bottom right
Model Photography	Indicates whether the photo is en-face / profile.	Integer/Numeric	1 to 2: en-face / profie
Price	Price of the product in USD.	Integer/Numeric	18 to 82
Price 2	Indicates whether the price of a specific product is higher than the entire product category.	Integer/Numeric	1 to 2: YES / No
Page	Page number of the shopping website.	Integer/Numeric	1 to 5
Price Category	Potential amount a user might spend in the current session.	String	0 - 25: Budget, 25 - 35: Value, 35 - 65: Average, 65 - 100: Premium

Phase 2: Exploratory Data Analysis (EDA) and Feature Selection



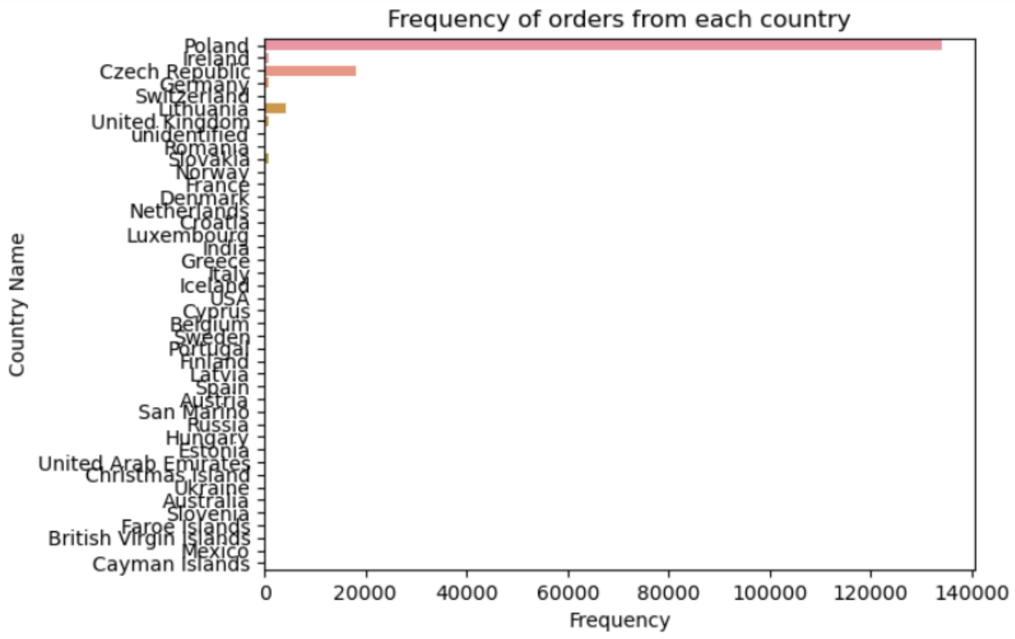
Mean pricing is about 44 & Median is about 43





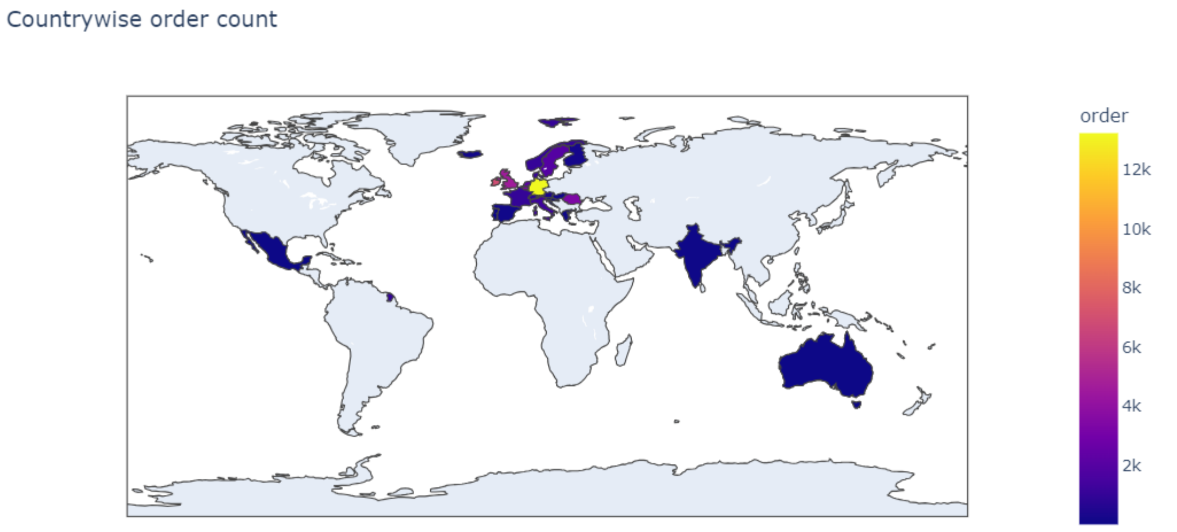
Clearly, the dates with lighter colors have highest number of orderes recieved. Rest seem to be recieve average number of orders. The dates with black or darker shades of green / blue have minimum number of orders. The following is observed:

- April has major number of orders (More lighter color cells)
- June has lowest amount of orders recieved
- May & July almost have equal amounts of orders recieved.
- Aug has one day with very high amounts of orderes recieved.

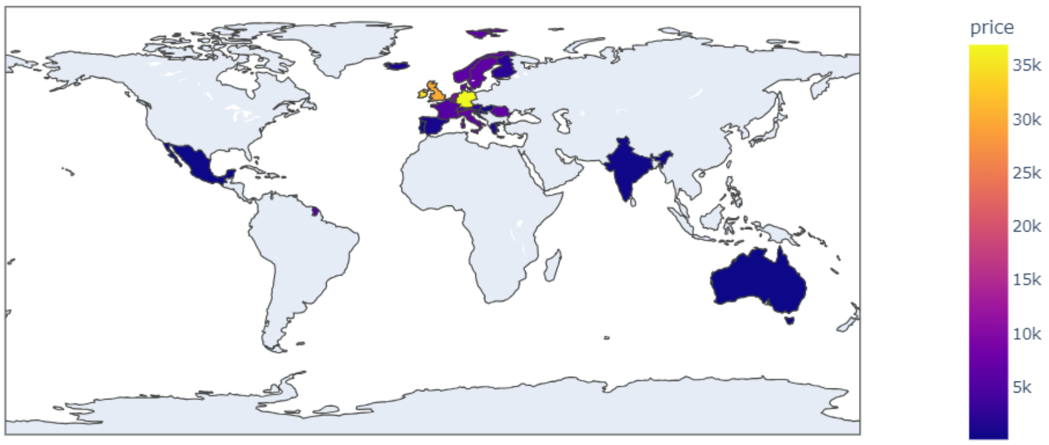


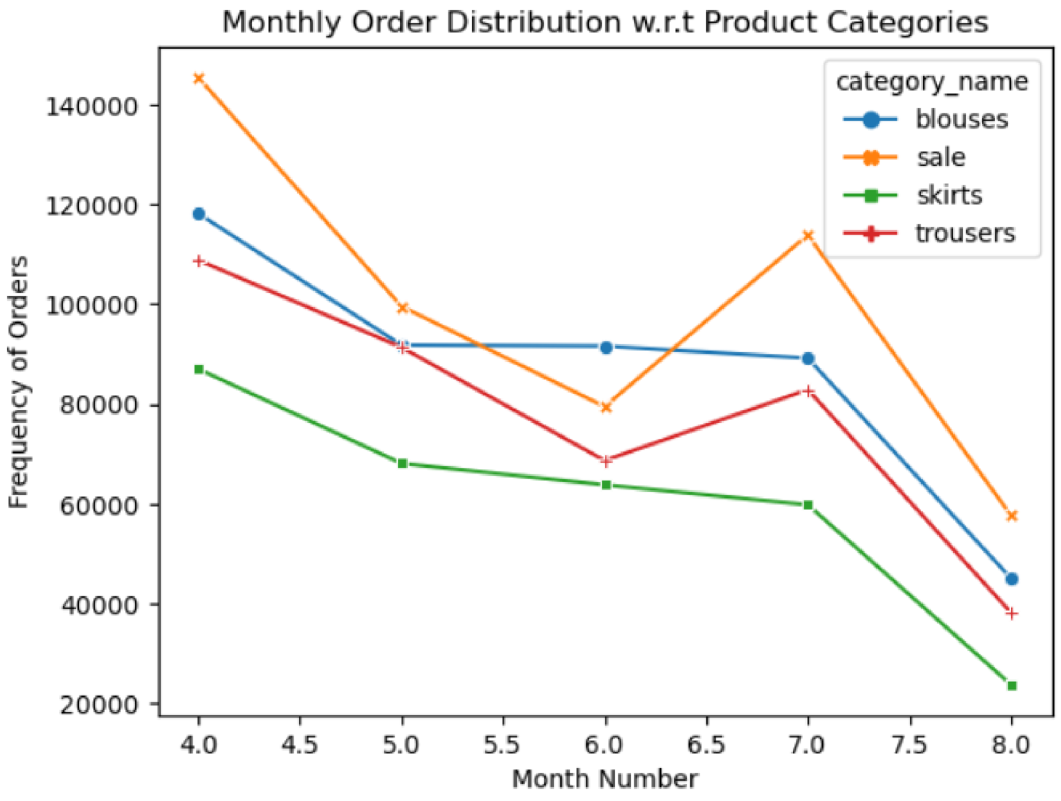
Clearly, countries like Poland, Czech Republic, and Lithuania have highest frequency of orders.

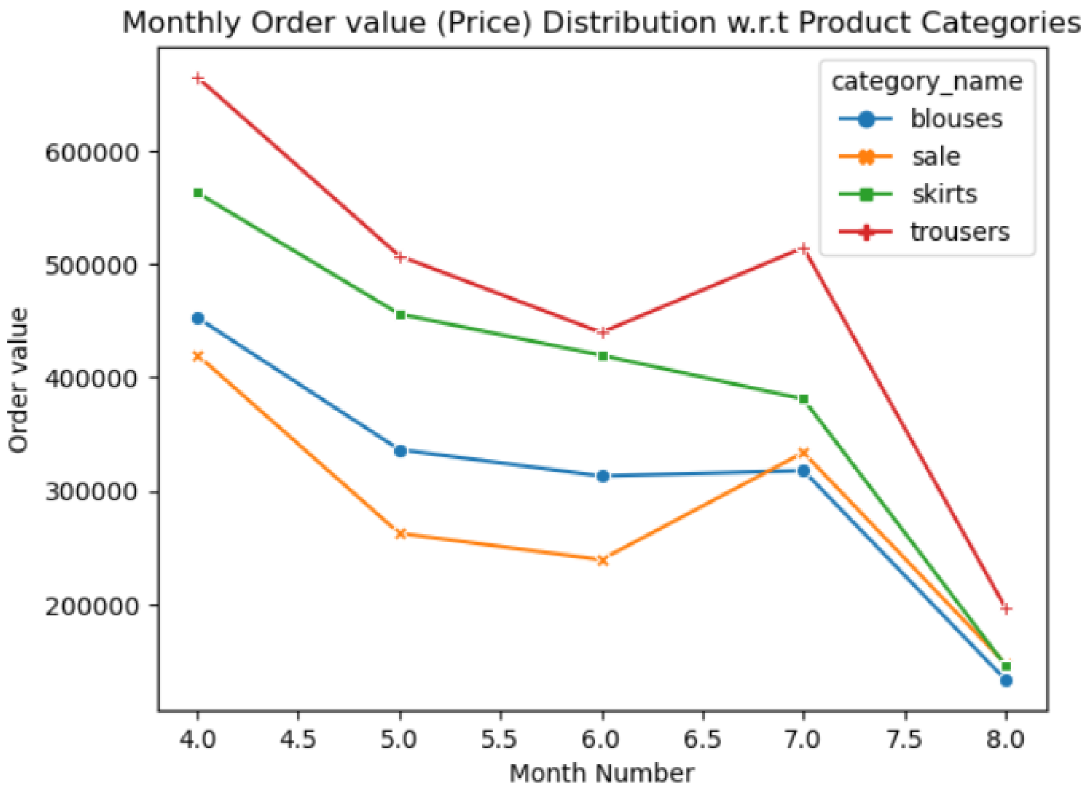
We shall remove these countries of very high frequency and check the order frequency from other countries.



Price contribution from each country





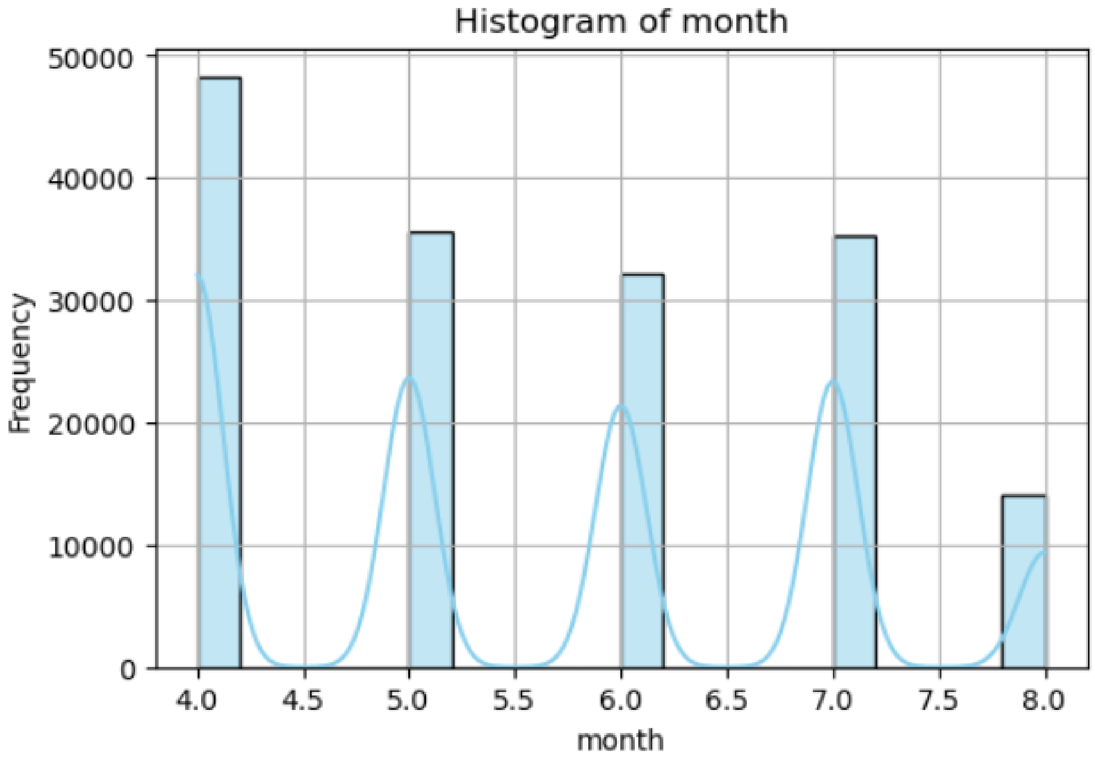


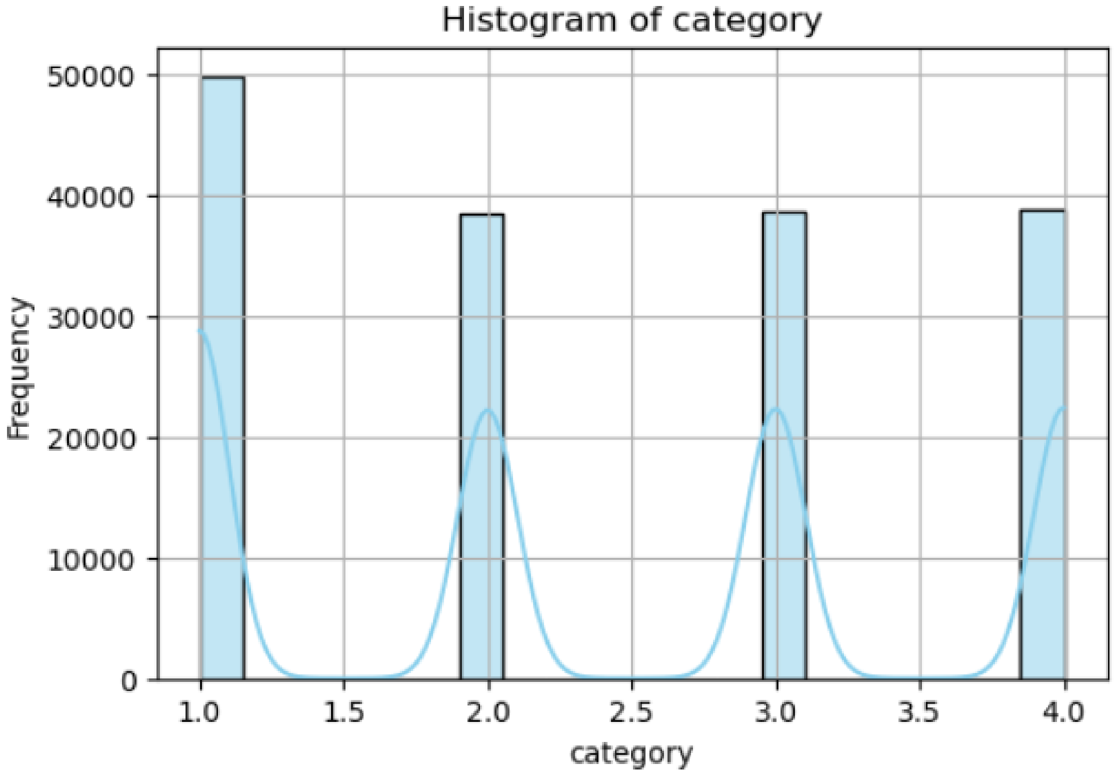
Data Preprocessing:

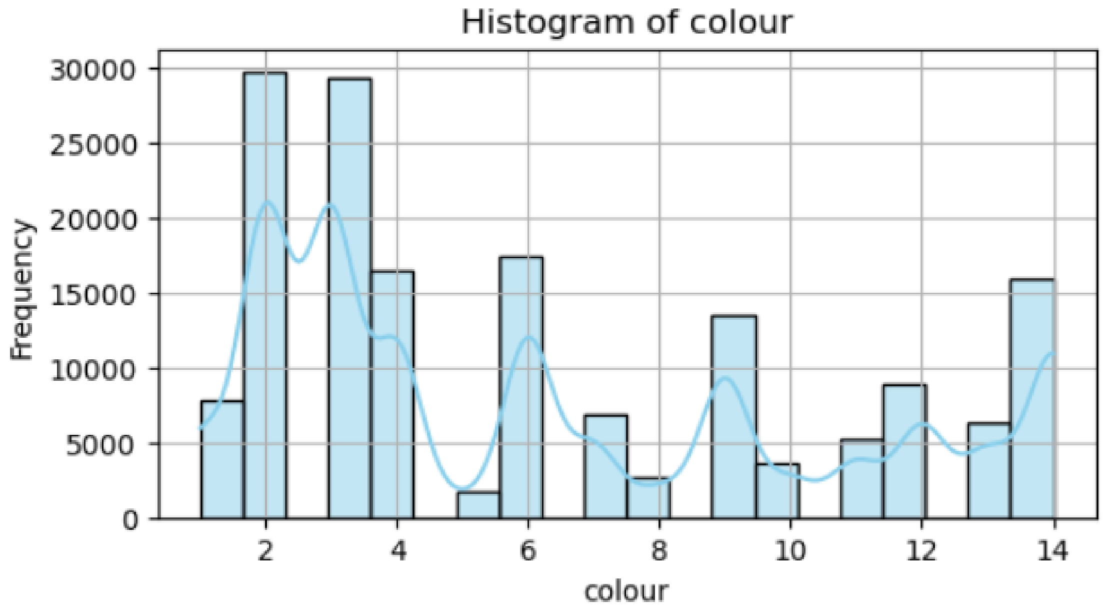
- Encoding categorical data to Numerical
- Data Scaling (Data Standardization)

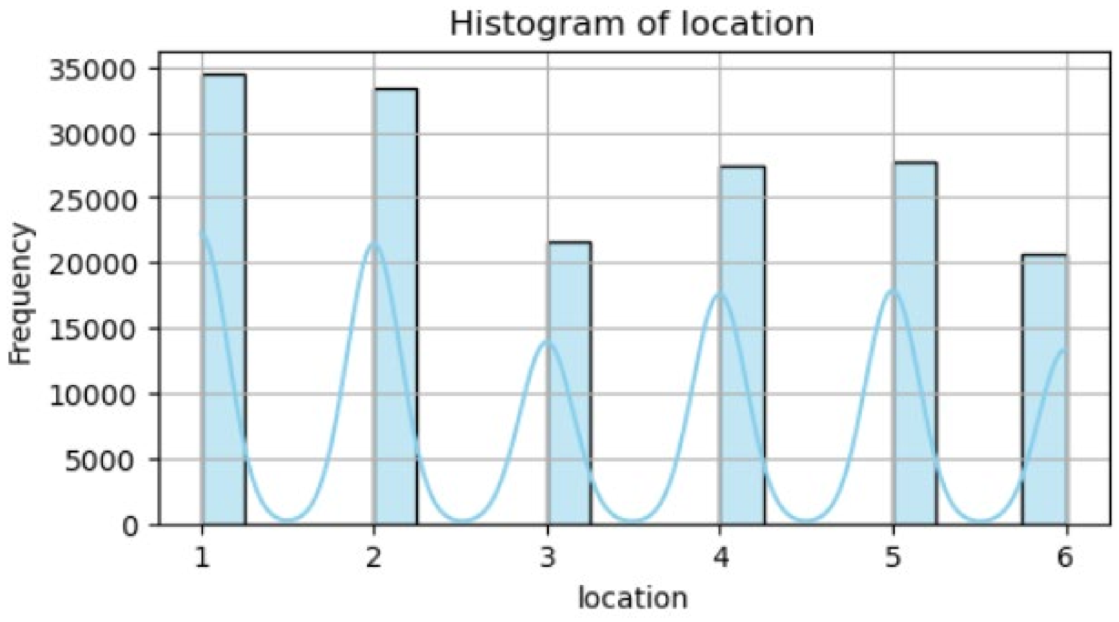
Feature Selection:

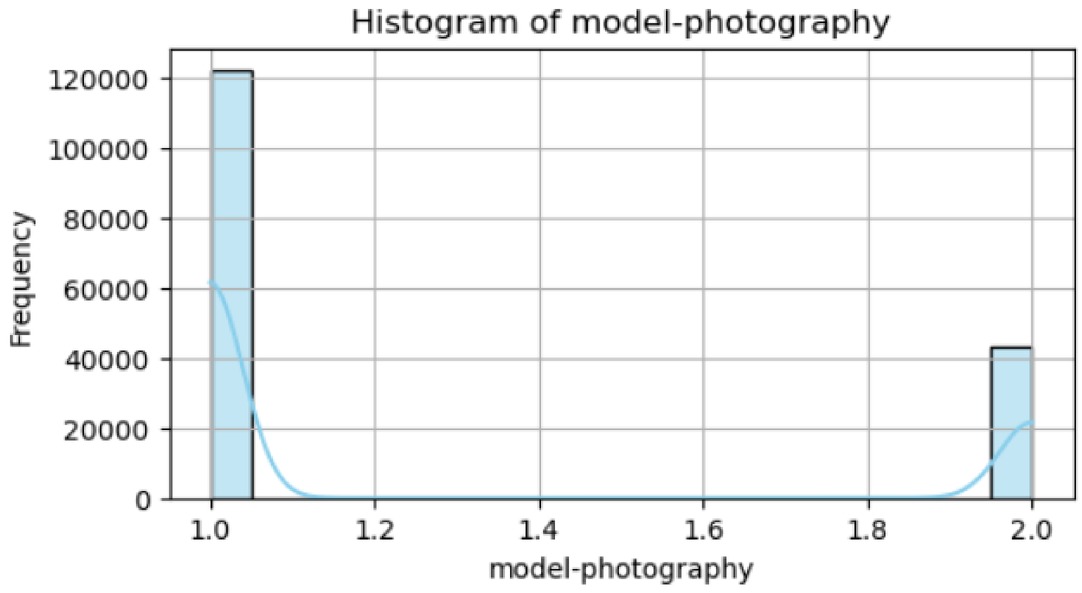
- Correlation Analysis
- Regression Analysis
- Has been performed using Random Forest Regressor and top 6 features with high importance are chosen for


Histogram of month
Frequency
month
0
10000
20000
30000
40000
50000
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Histogram of category
Frequency
category
0
10000
20000
30000
40000
50000
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Histogram of colour
Frequency
colour
0
5000
10000
15000
20000
25000
30000
2
4
6
8
10
12
14


Histogram of location
Frequency
location
0
5000
10000
15000
20000
25000
30000
35000
1
2
3
4
5
6



Data Split:

- Train / Valid / Test Split

<u>Phase 3: Model Implementation and Baseline Evaluation</u>

Model Implementation & Training:

Regression

- Linear Regression
- Ridge Regression
- LASSO Regression
- KNN Regressor
- Decision Tree Regressor
- Random Forest Regressor
- Bagging Regressor
- MLP Regressor

Classification

- Logistic Regression
- Gaussian Naive Bayes
- K-nearest Neighbors(KNN)

- Bagging
- Decision Tree Classifier
- Random Forest Classifier
- MLP Classifier

Model Benchmarking:

- Compare the results of each model across different models

Phase 4: Hyperparameter Tuning

Grid Search CV has been performed on Decision Tree Regressor and the MSE is found to be low

- param_grid**:** This dictionary stores the hyperparameters you want to explore.
  - Keys: Represent the names of the hyperparameters.
  - Values: Lists containing different values you want to try for each hyperparameter.
- max_depth**:** This hyperparameter controls the maximum depth of the decision tree, which influences its complexity. A deeper tree can capture more complex relationships but is prone to overfitting.
- min_samples_split**:** This hyperparameter defines the minimum number of samples required to split an internal node in the tree. Higher values can prevent overfitting by avoiding splitting nodes with too few data points.
- min_samples_leaf**:** This hyperparameter determines the minimum number of samples allowed in a leaf node (terminal node) of the tree. Larger values can prevent overfitting by ensuring leaves have enough data points to represent reliable decisions.
- scoring='neg_mean_squared_error'**:** This argument specifies the metric used to evaluate model performance. Here, neg_mean_squared_error (negative mean squared error) is used, which is typically appropriate for regression tasks where you want to minimize the squared difference between predicted and actual values.
- cv=5**:** This argument defines the number of folds used in cross-validation (CV). CV helps assess model performance on unseen data by splitting the training data into folds, training on some folds, and evaluating on others. Here, 5-fold CV is used.
- return_train_score=True**:** This argument instructs the tuning process to also return training scores alongside the cross-validation scores.

Hyperparameter Tuning for Other Models:

While the specific hyperparameters may differ, the general approach to hyperparameter tuning remains similar across various machine learning models. Here are some common models and their key hyperparameters:

1. Decision Tree (DecisionTreeClassifier for classification, DecisionTreeRegressor for regression):

1. **max_depth:** This hyperparameter controls the maximum depth a decision tree can grow to. A deeper tree can capture more complex relationships in the data but is also more prone to overfitting. Higher values often lead to better performance on the training data but might not generalize well to unseen data.
2. **min_samples_split:** This hyperparameter defines the minimum number of samples required to split an internal node in the tree. Higher values can prevent overfitting by avoiding splitting nodes with too few data points. If a node has less than min_samples_split samples, it becomes a leaf node (terminal node) without further splitting.
3. **min_samples_leaf:** This hyperparameter determines the minimum number of samples allowed in a leaf node (terminal node) of the tree. Larger values can help prevent overfitting by ensuring leaves have enough data points to represent reliable decisions. If a node has fewer samples than min_samples_leaf, it might be merged with a sibling node to reach the minimum requirement.
4. **criterion:** This hyperparameter specifies the function used to measure the quality of a split at a node in the tree. Common options are:
   - **"gini" (Gini impurity):** This metric is often used for classification tasks. It measures the degree of homogeneity or purity within a node. A lower Gini impurity score indicates a more homogeneous node (better split).
   - **"entropy" (Information gain):** This metric is also used for classification tasks. It measures the decrease in information entropy (uncertainty) after a split. A higher information gain (greater decrease in entropy) signifies a better split.
   - **"squared_error" (Mean squared error):** This metric is commonly used for regression tasks. It measures the average squared difference between predicted and actual values. A lower squared error indicates a better split, leading to a more accurate model.

How These Hyperparameters Affect Model Behavior:

- **Balancing Complexity and Overfitting:** max_depth and min_samples_split and min_samples_leaf work together to control the complexity of the decision tree.
  - Lower max_depth and higher min_samples_split and min_samples_leaf values lead to shallower trees with less risk of overfitting but might not capture complex patterns.
  - Higher max_depth and lower min_samples_split and min_samples_leaf values might lead to deeper trees that can be overfit on the training data.
- **Splitting Criteria:** The choice of criterion influences how the decision tree selects the best split at each node.
  - **"gini" or "entropy"** are suitable for classification tasks, helping the tree create purer leaves with better class separation.
  - **"squared_error"** is appropriate for regression tasks, guiding the tree towards splits that minimize the average squared error between predicted and actual values.

2. Random Forest (RandomForestClassifier, RandomForestRegressor):

- n_estimators**:** The number of decision trees to build in the forest. More trees generally improve performance but increase training time.
- max_depth**:** Similar to decision trees, controls the maximum depth of individual trees.
- min_samples_split**:** Similar to decision trees.
- min_samples_leaf**:** Similar to decision trees.

3. Support Vector Machine (SVC, SVR):

- C**:** Regularization parameter that controls the trade-off between fitting the training data and avoiding overfitting.
- kernel**:** The kernel function used to transform data into a higher-dimensional space for non-linear decision boundaries (e.g., "linear", "poly", "rbf").

4. K-Nearest Neighbors (KNeighborsClassifier, KNeighborsRegressor):

- n_neighbors**:** The number of neighbors to consider for classification or prediction.
- metric**:** The distance metric used to calculate distances between samples (e.g., "euclidean", "manhattan", "minkowski").

5. Naive Bayes (GaussianNB, MultinomialNB, BernoulliNB):

- var_smoothing**:** A parameter used to add a small amount of smoothing to variance estimates, which can help prevent numerical instability (especially for GaussianNB).

6. Neural Networks (MLPClassifier, MLPRegressor):

- hidden_layer_sizes**:** A tuple defining the number of neurons in each hidden layer (e.g., (100, 50) for two hidden layers with 100 and 50 neurons,

Phase 5: Model Evaluation and Comparative Analysis

Regression

Algorithm	R^2 – Model Fit	MSE - Test
Linear	0.16747596662357034	132.91989161662835
Ridge	0.1674754699621897	132.9199709130501
LASSO	0.0000	159.66079623885835
KNN	0.9999859086712255	0.0007856646420693198
Decision Tree	1.0	0.0
Random Forest	0.9999985333509067	5.3452391744477544e-05
Bagging	0.868052351525576	0.00010908651376424
Neural Network (MLP)	0.872126888544669	17.18390888037176

Mean Squared Error after Hyperparameter Tuning: 24.38133892046599

Algorithm	**Accuracy**
Logistic	0.69
Gaussian Naïve Bayes	1.000
KNN	0.9999859086712255
Decision Tree	1.0
Random Forest	0.9999985333509067
Bagging	0.868052351525576
MLP Classifier	0.989

Regression Model Evaluations:

- **Linear Regression:**
  The linear regression model has a low R^2 value, indicating that it explains only a small portion of the variance in the data. The MSE is also relatively high.
- **Ridge Regression:**
  Ridge regression doesn't seem to improve upon the performance of linear regression significantly in this case.
- **LASSO Regression:**
  LASSO regression does not perform well compared to other models, with an R^2 of 0 and a relatively high MSE.
- **K-Nearest Neighbors (KNN):**
  KNN performs exceptionally well with a very high R^2 value and a very low MSE, suggesting it's overfitting the training data.
- **Decision Tree:**
  The decision tree model seems to perfectly fit the training data, achieving an R^2 of 1 and a MSE of 0 on the test data.
- **Random Forest:**
  Random forest performs almost as well as the decision tree with very high R^2 and low MSE values.
- **Bagging:**
  Bagging performs well but not as well as decision tree or random forest.
- **Neural Network (MLP):**
  The neural network (MLP) performs reasonably well with a high R^2 value but a relatively higher MSE compared to some other models.

Comparative Analysis of Regression Models:

- Among the regression models, KNN, decision tree, and random forest stand out as top performers with very high R^2 values and very low MSE.
- While neural networks (MLP) perform reasonably well, they have a higher MSE compared to tree-based methods.
- Linear regression, ridge regression, and LASSO regression perform poorly compared to other models, indicating that the data may not be well-suited for linear methods.
- Decision tree and random forest appear to be the best-performing models overall, followed closely by KNN. However, decision trees might be prone to overfitting.
-

Classification Model Evaluations:

- **Logistic Regression:**
  Logistic regression achieves moderate accuracy but seems to underperform compared to other models in this scenario.
- **Gaussian Naïve Bayes:**
  Gaussian Naïve Bayes achieves perfect accuracy on the dataset, indicating that the features are well-separated according to class labels.
- **K-Nearest Neighbors (KNN):**
  KNN performs exceptionally well with almost perfect accuracy, indicating that instances with similar features tend to belong to the same class.
- **Decision Tree:**
  Decision tree achieves perfect accuracy on the dataset, suggesting that it's able to perfectly partition the feature space based on the class labels.
- **Random Forest:**
  Random forest performs almost as well as the decision tree with nearly perfect accuracy.
- **Bagging:**
  Bagging achieves lower accuracy compared to other ensemble methods like random forest and decision tree.
- **MLP Classifier (Neural Network):**
  The MLP classifier performs well with high accuracy but slightly lower than some other models.

Comparative Analysis of Classification Models:

- Gaussian Naïve Bayes, decision tree, and random forest achieved perfect or near-perfect accuracy on the dataset, indicating that they are able to capture the underlying patterns in the data very well.
- KNN also performed exceptionally well, with very high accuracy.
- MLP classifier achieved high accuracy but slightly lower compared to the top-performing models.
- Logistic regression and bagging performed relatively less accurately compared to other models in this scenario.
- In conclusion, for this particular dataset, Gaussian Naïve Bayes, decision tree, random forest, KNN, and MLP classifier appear to be the top-performing models for classification tasks, achieving high accuracy scores.

Phase 6: Conclusion and Recommendations

Conclusion and Recommendations:

After conducting a comprehensive analysis of various classification models, including Gaussian Naïve Bayes, decision tree, random forest, KNN, MLP classifier, logistic regression, and bagging, several key insights have emerged.

Key Findings:

- Gaussian Naïve Bayes, decision tree, random forest, KNN, and MLP classifier demonstrated excellent performance in terms of accuracy, with some achieving perfect or near-perfect scores. These models were able to effectively capture the underlying patterns in the dataset.
- Logistic regression and bagging, while still performing reasonably well, showed comparatively lower accuracy rates in this scenario.
- The choice of the appropriate classification model depends on various factors such as the nature of the dataset, computational resources, interpretability requirements, and the specific goals of the analysis.

Interpretation and Inference:

- The high accuracy scores achieved by the top-performing models indicate that they are well-suited for making accurate predictions on the given dataset.
- Gaussian Naïve Bayes assumes independence among features given the class label, which may or may not hold true in real-world scenarios. However, its simplicity and effectiveness make it a popular choice for classification tasks, especially with smaller datasets.
- Decision trees and random forests offer interpretability, as they can be visualized to understand how the model makes decisions. This can provide valuable insights into the underlying relationships between features and the target variable.
- KNN's performance suggests that instances with similar feature values tend to belong to the same class, making it a suitable choice for datasets where instances of the same class are close to each other in the feature space.
- MLP classifiers, being neural network-based models, offer flexibility and can capture complex nonlinear relationships in the data. However, they may require more computational resources for training and may not be as interpretable as simpler models like decision trees.

Recommendations:

- Based on the analysis, it is recommended to further investigate the dataset to gain deeper insights into the characteristics of the features and their relationships with the target variable.
- Depending on the specific requirements of the classification task, one or more of the top-performing models (Gaussian Naïve Bayes, decision tree, random forest, KNN, MLP classifier) can be selected for deployment.

- It is advisable to evaluate the models on additional metrics such as precision, recall, and F1-score to gain a more comprehensive understanding of their performance, especially in scenarios with class imbalance.
- Regular monitoring and updating of the deployed model(s) are recommended to ensure continued accuracy and relevance, especially in dynamic environments where the data distribution may change over time.

In conclusion, the analysis provides valuable insights into the performance of various classification models, facilitating informed decision-making in selecting the most suitable model(s) for the given dataset and task at hand.

“No Free Lunch theorem in machine learning states that no single machine learning algorithm is universally superior for all tasks”