Lab 5 Assignment Report

Assignment

CS503: Machine Learning

Name	Entry No.
Anshika	2021CSB1069

Instructor: Dr. Santosh Vipparthi

Teaching Assistant: Nishchala Thakur

Indian Institute of Technology Ropar
Punjab, India
February 22, 2025

Contents

1 | Overview

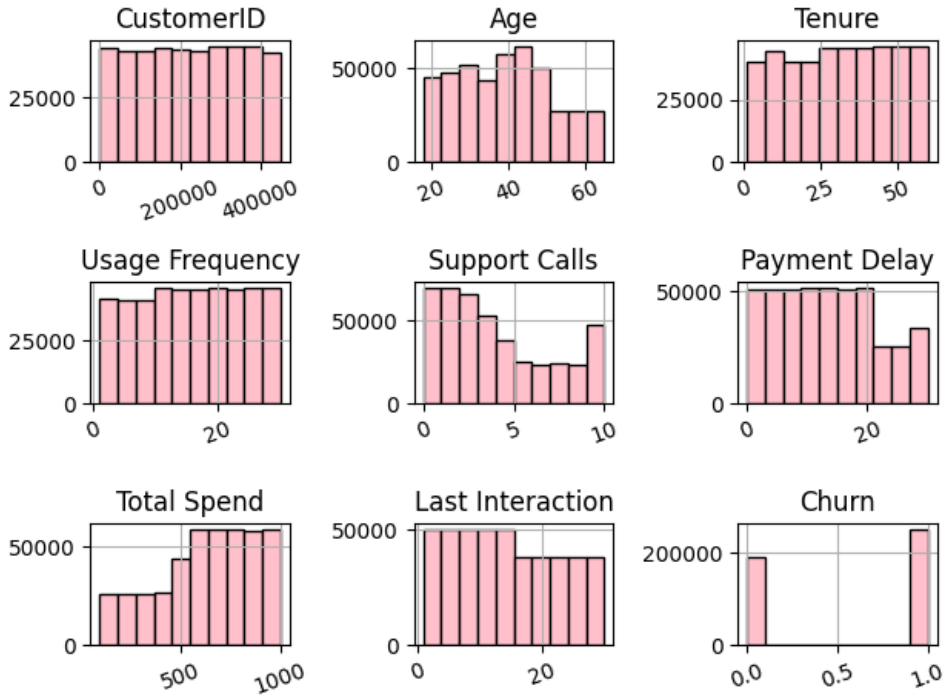
In this report an in-depth analysis of customer churn data is presented by detailing the key preprocessing steps and evaluation strategies used to develop an effective model for churn prediction. Systematic preprocessing techniques are applied to improve data quality, ensure appropriate transformations, and use suitable metrics to assess model performance. A robust foundation for the model's training and evaluation process is established through careful data exploration, visualisation, and preparation.

2 | Data Preprocessing

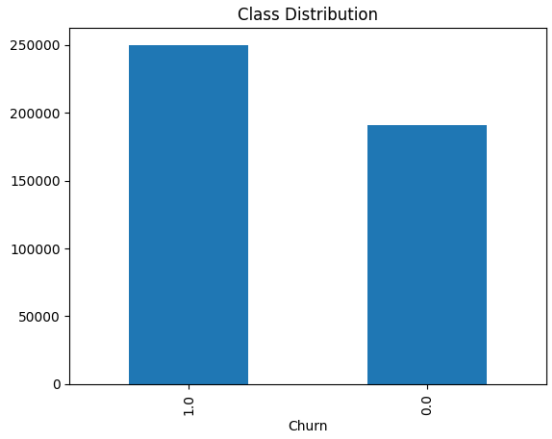
Following are the steps taken to ensure data quality and address any issues that could potentially affect model performance.

2.1 | Data Visualization

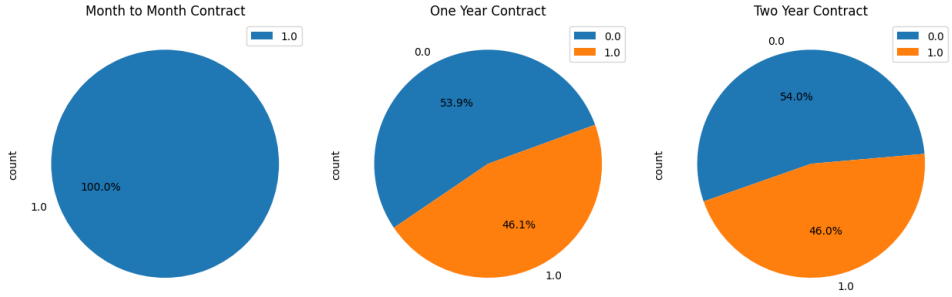
- **Histograms**: for the numeric columns, which helped identify the distribution and range of the features.



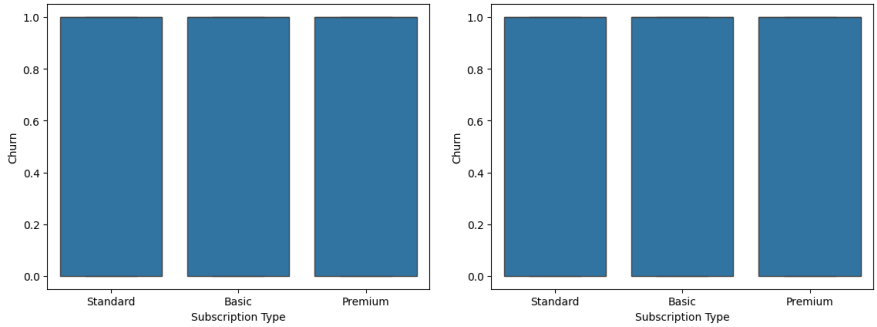
- **Class Distribution:** target variable 'Churn' was visualised to check the balance of the classes.



- **Pie Charts:** to analyze the churn rate for different contract lengths, such as monthly, quarterly, and annual contracts.



- **Boxplots:** to investigate the relationship between 'Subscription Type' and 'Churn', providing insights into the distribution of churn across different subscription types.



2.2 | EDA Steps

- **Data Overview:** the overall structure of the dataset including checking column names, data types, and the presence of missing values.

- **Shape and Description:** The dataset shape examined (i.e., the number of rows and columns), and summary statistics for each numeric column, including measures of central tendency, spread, and potential outliers.

- **Correlation Heatmap:** to understand the relationships between numeric features to identify highly correlated variables, which guided feature selection and engineering.

2.3 | Handling Missing Values

- **Imputation and Removal:** Missing values in the dataset were addressed. Columns with missing values, such as 'Age', were either imputed or removed, depending on their significance and the number of missing entries.

- **Null Value Percentage:** The percentage of missing values for each column was calculated, and necessary steps were taken to handle them to prevent issues during model training.

2.4 | Categorical Variable Encoding

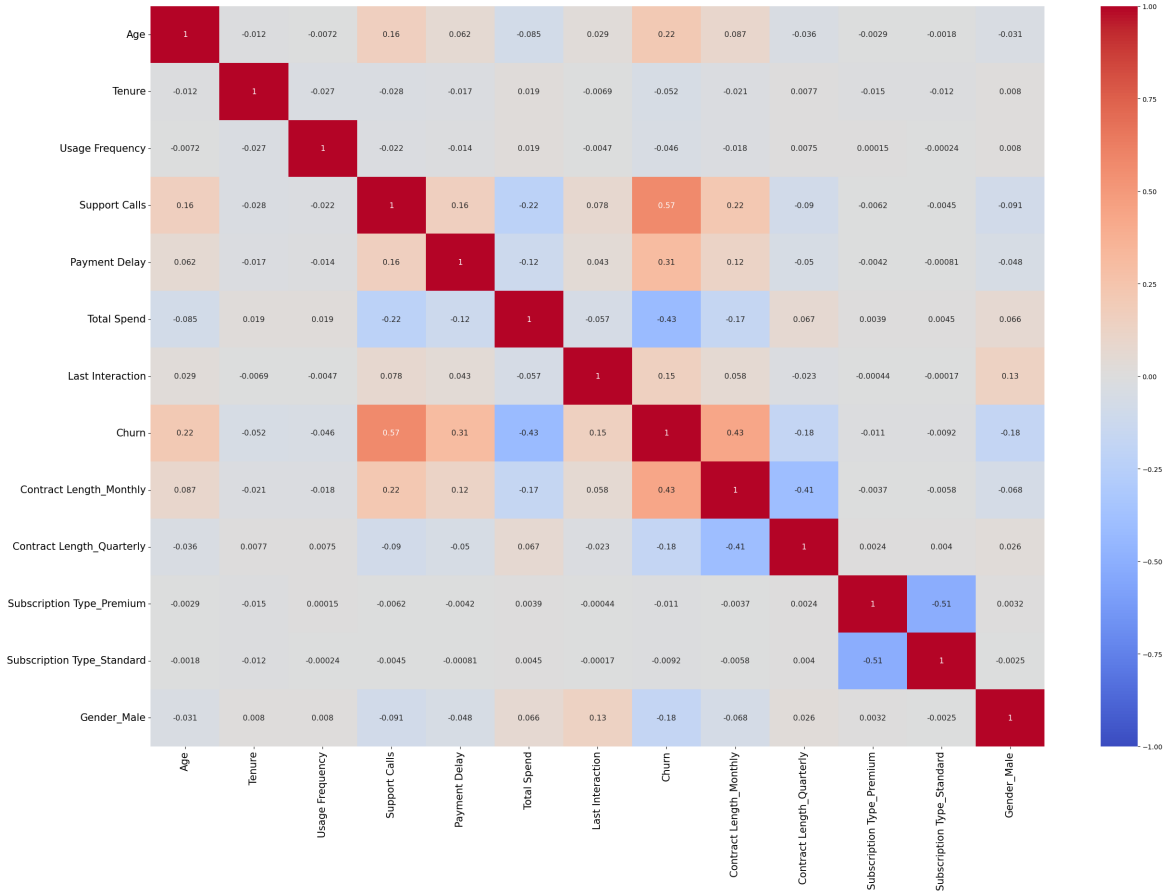
Several categorical features such as 'Contract Length', 'Subscription Type', and 'Gender' were encoded into numerical values using dummy variables. This process was necessary to convert non-numeric data into a format suitable for machine learning models. The **pd.get_dummies()** function was used for this task, creating one-hot encoded variables for the categories.

2.5 | Data Cleaning

- **Dropping Unnecessary Columns:** Columns that were deemed irrelevant or redundant, such as 'CustomerID', were dropped from the dataset.
- **Handling Outliers:** Boxplots for certain features, like 'Subscription Type', were examined to identify potential outliers that could skew the model's predictions.

3 | Correlation Analysis

Correlation matrix was computed to identify any multicollinearity between the features. The resulting heatmap provided valuable insights into the strength and direction of relationships between features and the target variable ('Churn'). Strongly Correlated Features: The heatmap revealed both positive and negative correlations between variables, guiding the feature selection process for the model. By following these preprocessing steps, the dataset was transformed into a suitable format for model training and evaluation, with minimal noise and an appropriate balance of relevant features.



3.1 | Model Evaluation

3.1.1 | Algorithm Considerations

- **Model Selection:** Logistic Regression
- **Gradient Descent Variants:** To optimize the performance of the model, we implemented and compared three variants of gradient descent:
- **Batch Gradient Descent:** Processes the entire dataset to compute the gradient and update weights.
- **Mini-Batch Gradient Descent:** Processes small random batches of data, introducing some randomness for faster convergence.

- **Stochastic Gradient Descent (SGD):** Updates the weights using a single training example at a time, resulting in frequent updates that help escape local minima but may cause more variance.
- **Hyperparameter Tuning:**

  Various learning rates (`alpha`) were tested for each variant of gradient descent. A grid search over the learning rates identified the best alpha for each method. Results show that each gradient descent method benefits from different optimal learning rates.
- **Feature Selection:**

  The features were scaled using `StandardScaler` to standardize them, improving computational efficiency and convergence. This step ensures that features with large magnitudes do not dominate the learning process, resulting in faster convergence.

3.1.2 | Cost Function

The Cost function for a logistic regression problem where the dependent variable is binary and is modelled using a sigmoid activation function. The log-likelihood maximisation approach was used, which aligns with minimising the negative log-likelihood for logistic regression. The cost function is defined as:

$$J(\beta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

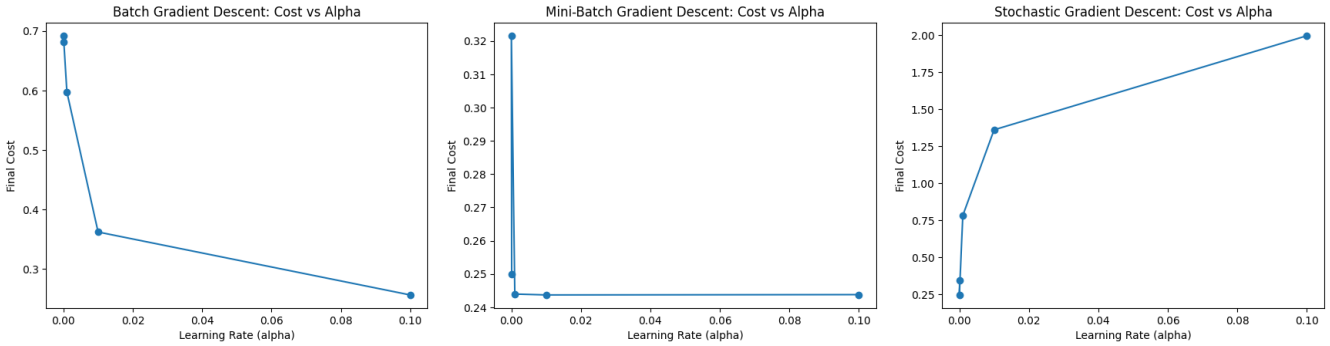
Where:

- y_i is the true label for the i-th sample.
- $\hat{y}_i = \frac{1}{1+e^{-z}}$ is the predicted probability using the sigmoid function, where $z = X \cdot \beta$.

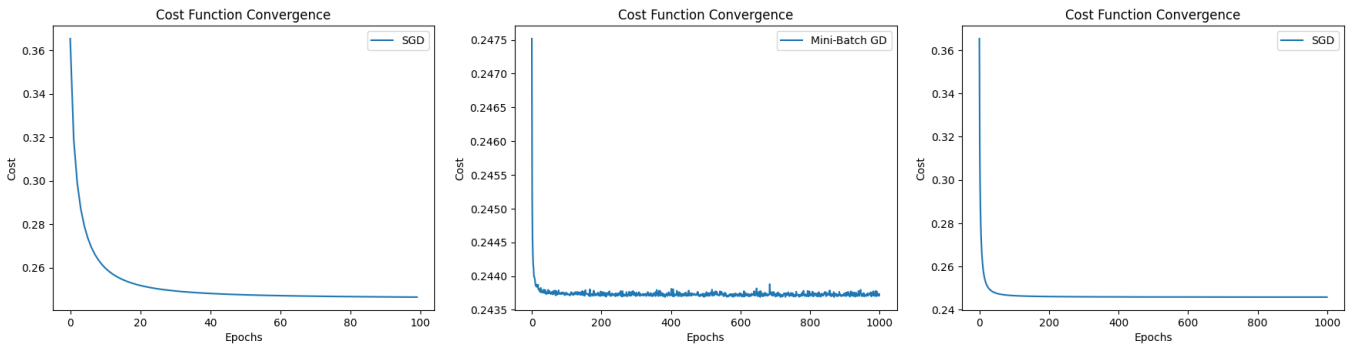
This cost function was minimized across all training samples to obtain the optimal model parameters β.

4 | Performance Measures

4.0.1 | Final cost vs Learning rate analysis for different learning rate 500 epochs



4.0.2 | Loss vs Learning rate analysis for 1000 epochs



4.1 | Performance Metrics

The performance metrics like **accuracy, precision, recall, F1 score, and ROC AUC** were compared for three variants of gradient descent: Batch Gradient Descent (BGD), Mini-Batch Gradient Descent (Mini-BGD), and Stochastic Gradient Descent (SGD).
Optimal Model Configuration: The hyperparameter learning rate is selected by careful tuning and analysis.
Visualisation:

- Confusion matrices were plotted to display the true positives, false positives, true negatives, and false negatives for each gradient descent variant, highlighting the effectiveness of each in classifying the samples.
- ROC curves were plotted to compare the trade-off between sensitivity and specificity for each model.
- Error distributions were plotted to observe the spread of prediction errors and identify any potential bias or variance issues.

4.2 | Steps followed in algorithm execution

The following steps were taken in the execution of the algorithm:

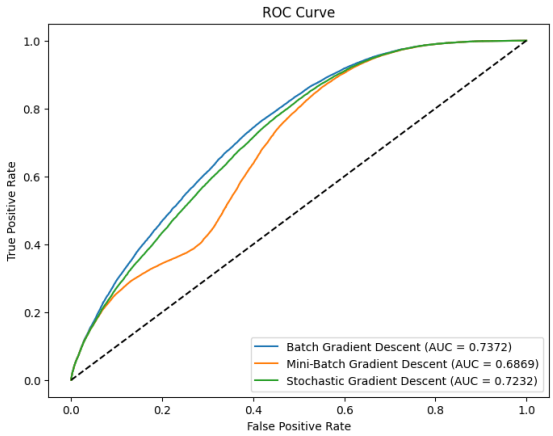
1. **Data Preprocessing:** As described earlier, the dataset was cleaned, standardized, and split into training and test sets. StandardScaler was used to scale the features, ensuring each feature contributed equally to the model training process.
2. **Model Training with Different Data Splits:** The three gradient descent variants (BGD, Mini-BGD, and SGD) were trained on the data using different training-to-test ratios. Each model was tuned with grid search to find the best learning rate for optimal performance.
3. **Performance Evaluation:** The models were evaluated based on a range of performance metrics (accuracy, precision, recall, F1 score, and ROC AUC). These metrics were then used to compare the effectiveness of each gradient descent variant in classifying the test data.
4. **Result Interpretation:** The analysis revealed that Stochastic Gradient Descent (SGD) had the best generalization performance, followed by Batch Gradient Descent (BGD) and Mini-Batch Gradient Descent (Mini-BGD). The higher variance in SGD allowed it to escape local minima and converge to a better solution, especially in more complex datasets.

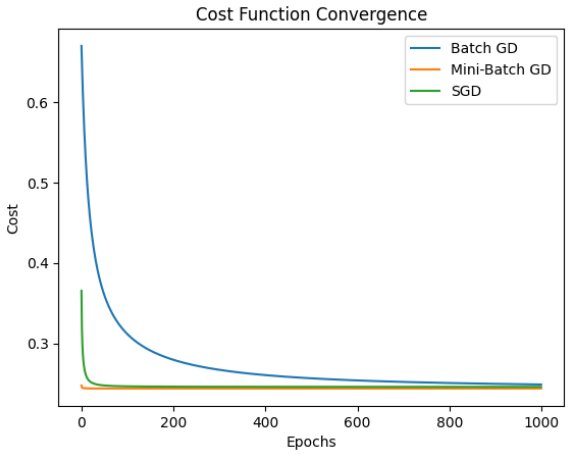
4.3 | Performance Metrics table for Testing

Table 4.1: Metrics for Gradient Descent Variants

Metrics	BGD	Mini-BGD	SGD
Accuracy	0.5738	0.5712	0.5804
Precision	0.5267	0.5251	0.5307
Recall	0.9899	0.9906	0.9878
F1 Score	0.6875	0.6864	0.6904
ROC AUC	0.5946	0.5921	0.6008
Confusion Matrix	[6752, 27129 308, 30185]	[6562, 27319 287, 30206]	[7244, 26637 373, 30120]

4.4 | ROC Curve plot





5 | Conclusion

During finding optimal hyperparameter learning rate for each algorithm, we observe Best alpha for Batch Gradient Descent: 0.1, Best alpha for Mini-Batch Gradient Descent: 0.01, Best alpha for Stochastic Gradient Descent: 0.00001. Therefore, for SGD, we require very low learning rate to bring down the loss. During training with optimal learning rate, we observe that Mini-BGD has lower cost throught and converges after 100 epochs, whereas BGD converges much later at around 900 epochs. SGD converges relatively earlier than Batch, as expected. Based on AUC, BGD shows best performance with 73.72% , SGD follows by 72.32% and Mini BGD has only 68.69%.