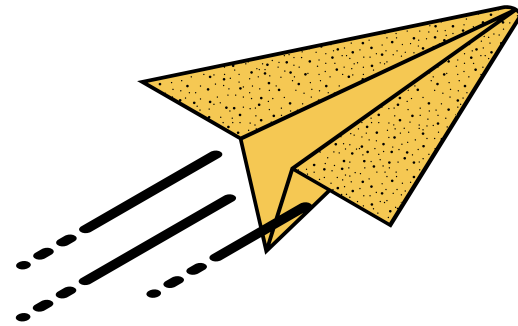
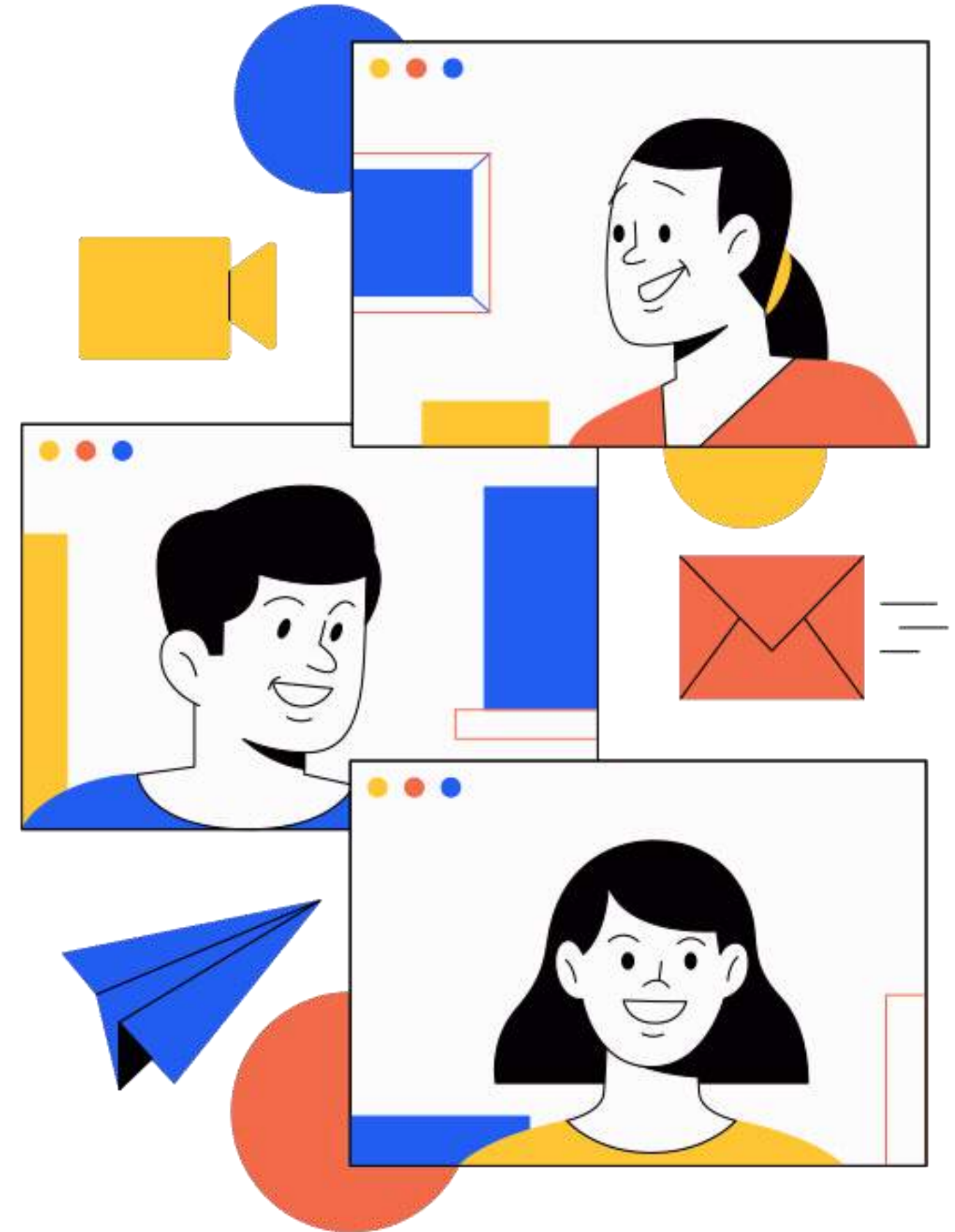


Presented by Group 3



CHURN MODELLING

Identifying likely churners to prevent revenue loss.



THE TEAM



Naveen Kumar



Aneesh Sharma



**Jitendra Kumar
Singh**



**Thanga Saravana
Kumar**



Vishnu N

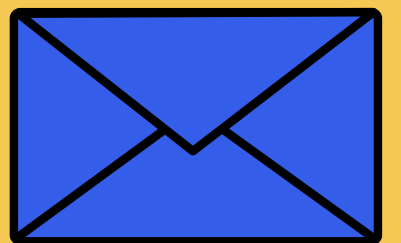
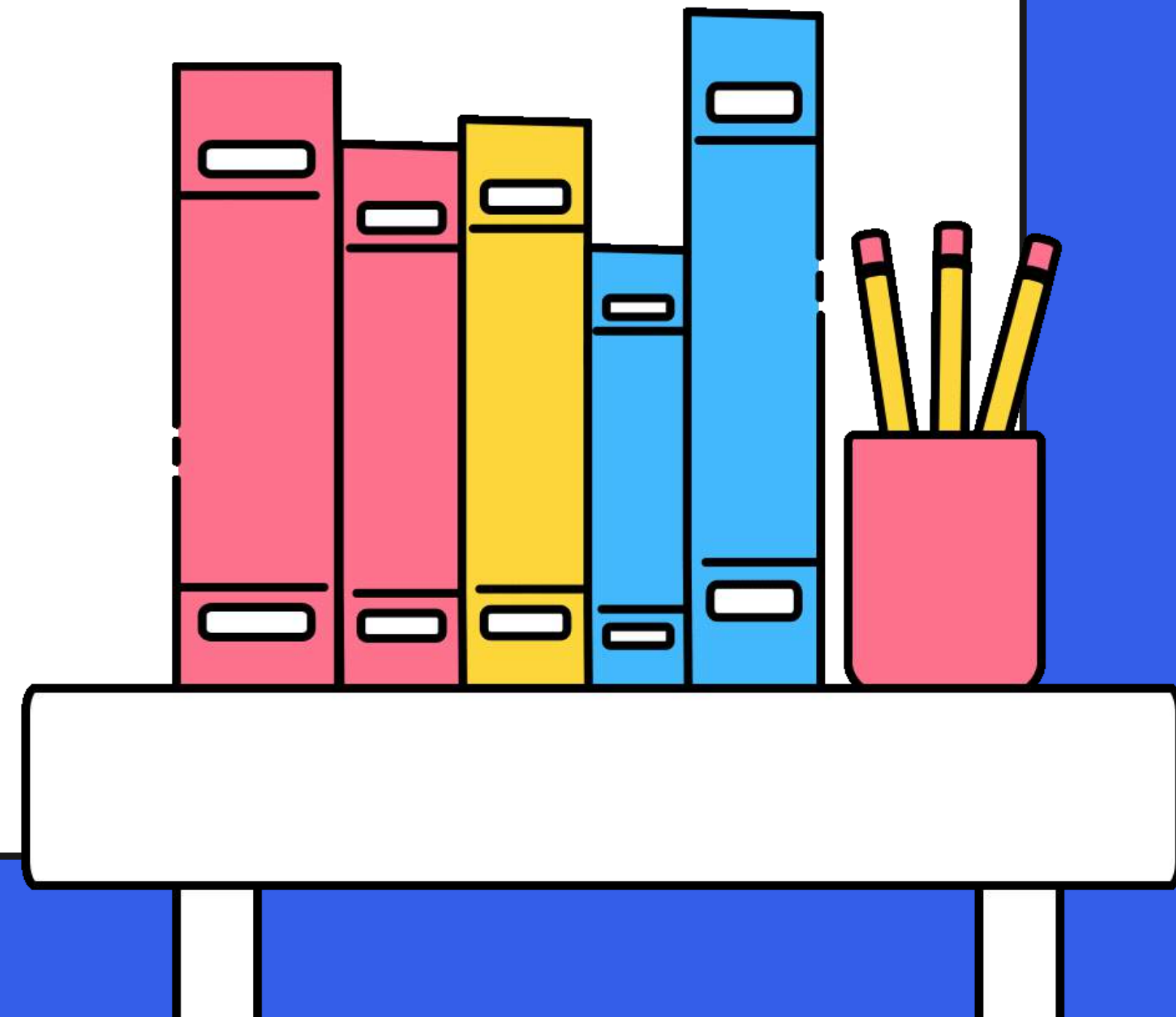
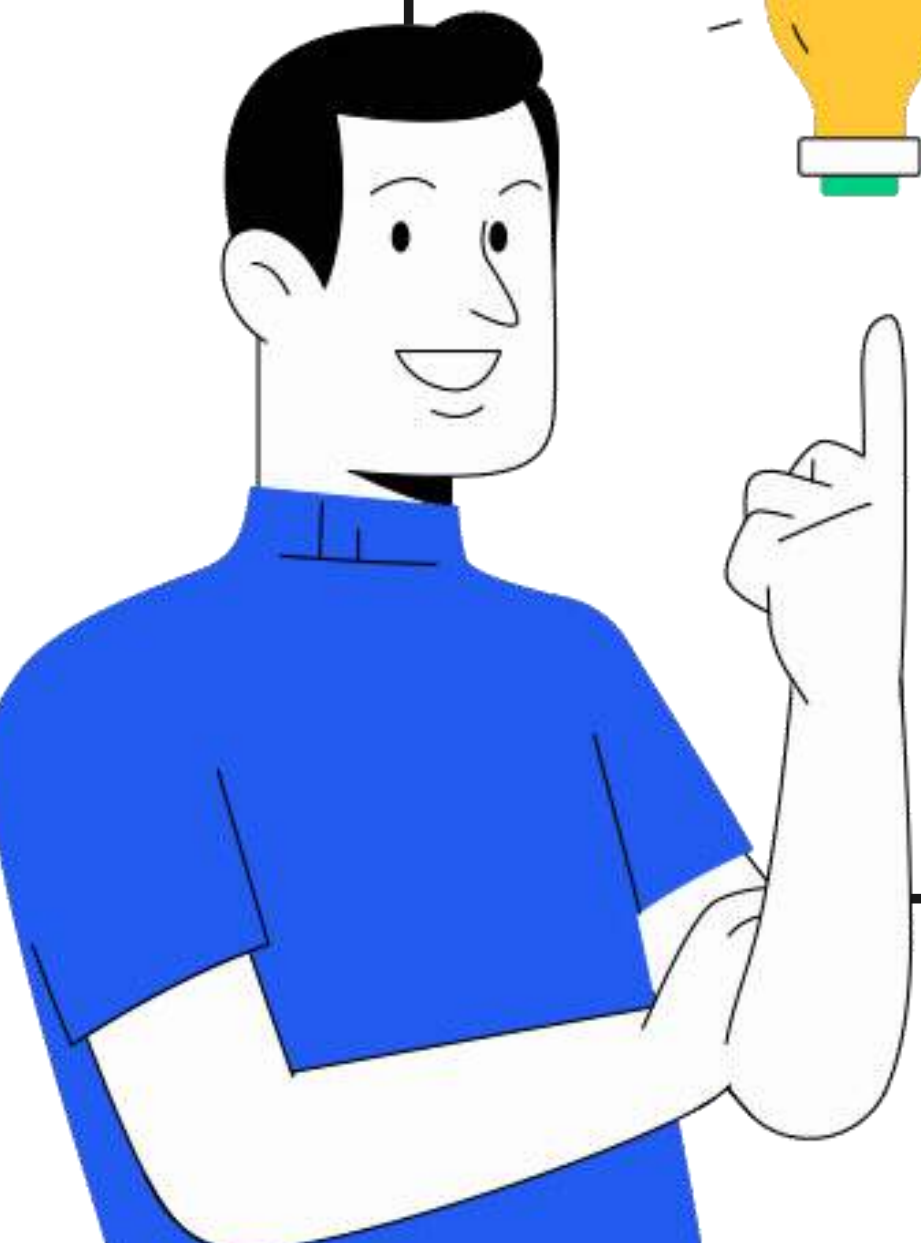


TABLE OF CONTENTS

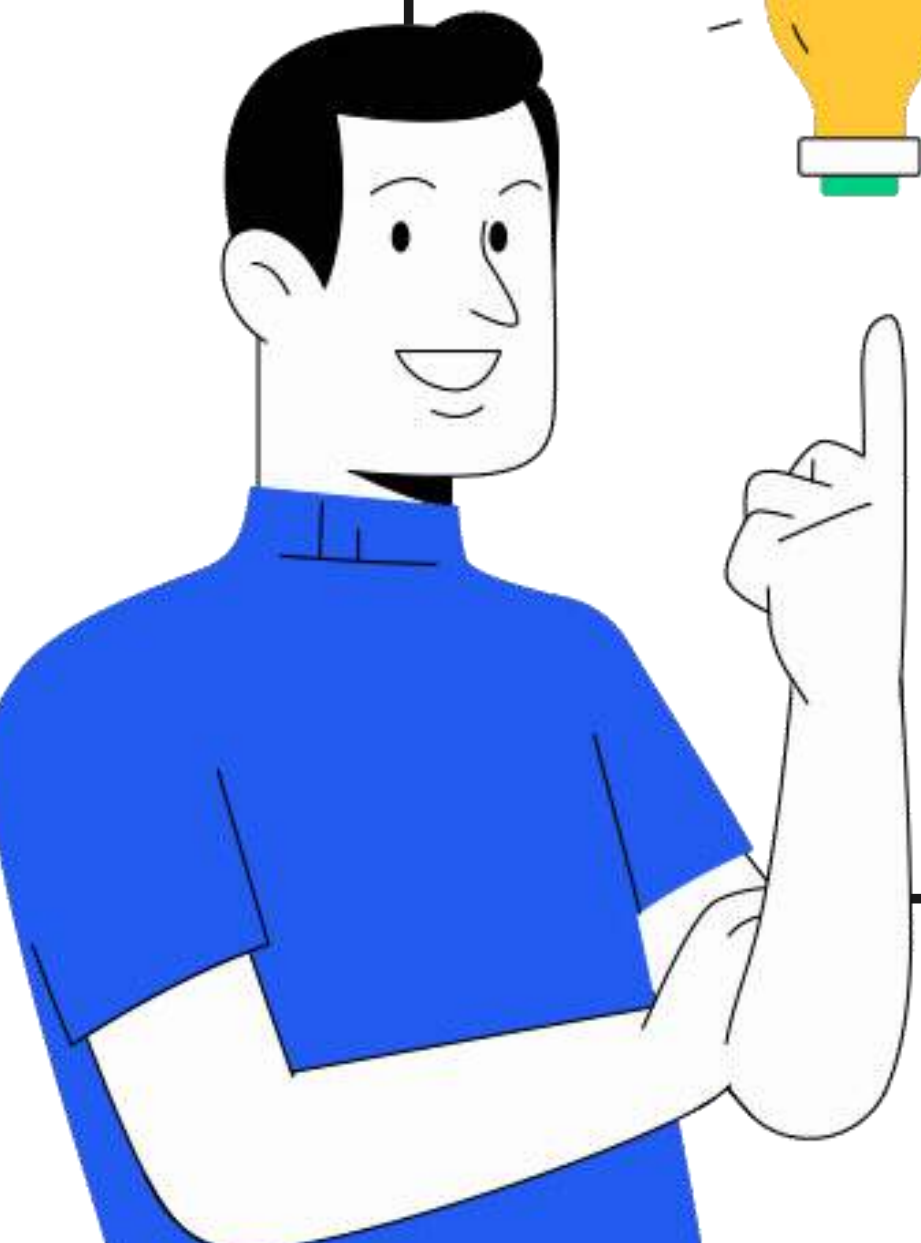
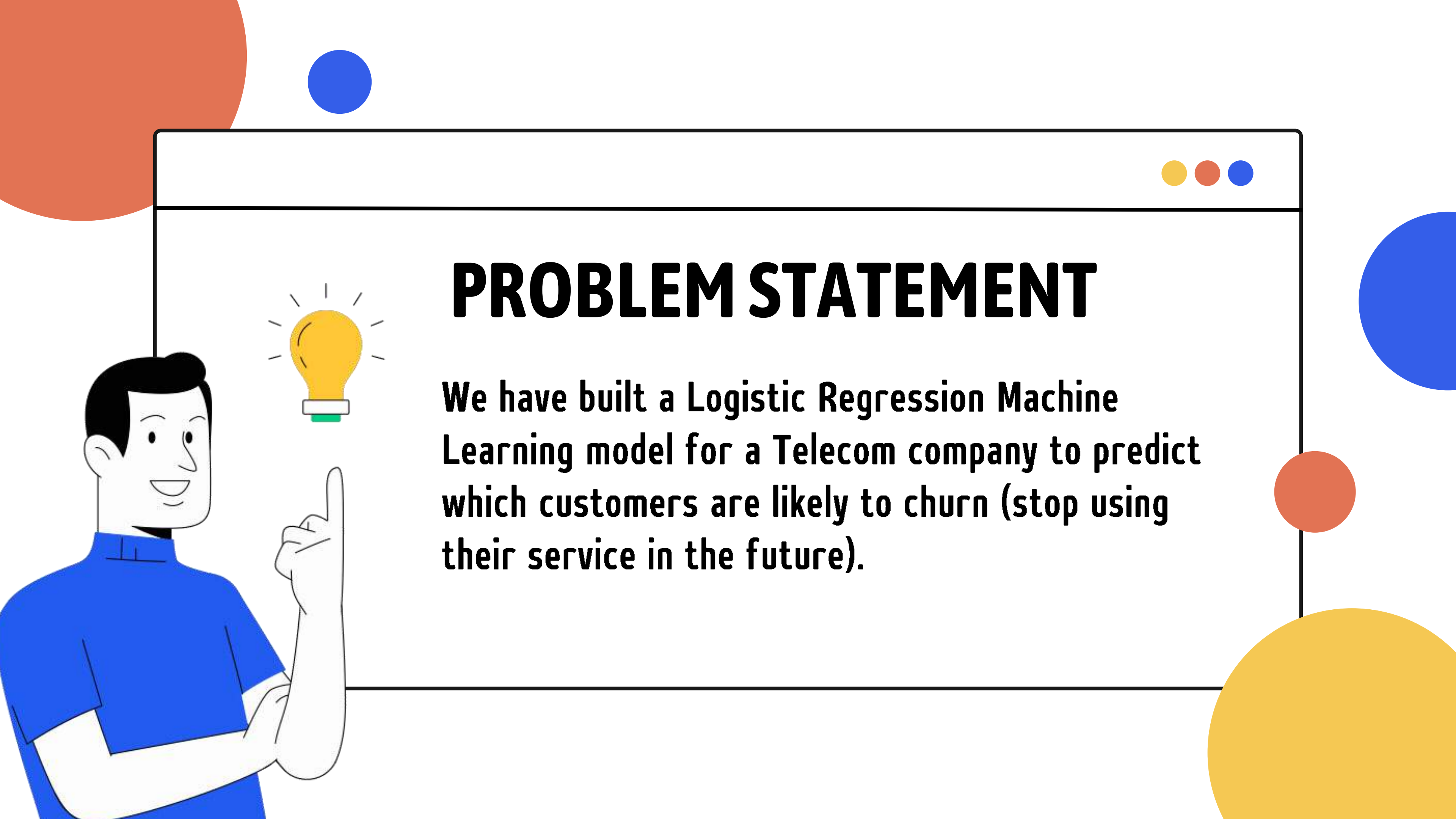
1. BUSINESS OBJECTIVE
2. PROBLEM STATEMENT
3. ANALYSING THE DATASET
4. FEATURE TRANSFORMATION
5. HEAT MAP FOR TUNED TRANSFORMED VARIABLES
6. LABEL ENCODING & SMOTE
7. BUILD AND EVALUATE MODELS
8. HYPER PARAMETER TUNING USING GRID SEARCH
9. SHAP(SHAPELY ADDITIVE EXPLANATIONS)





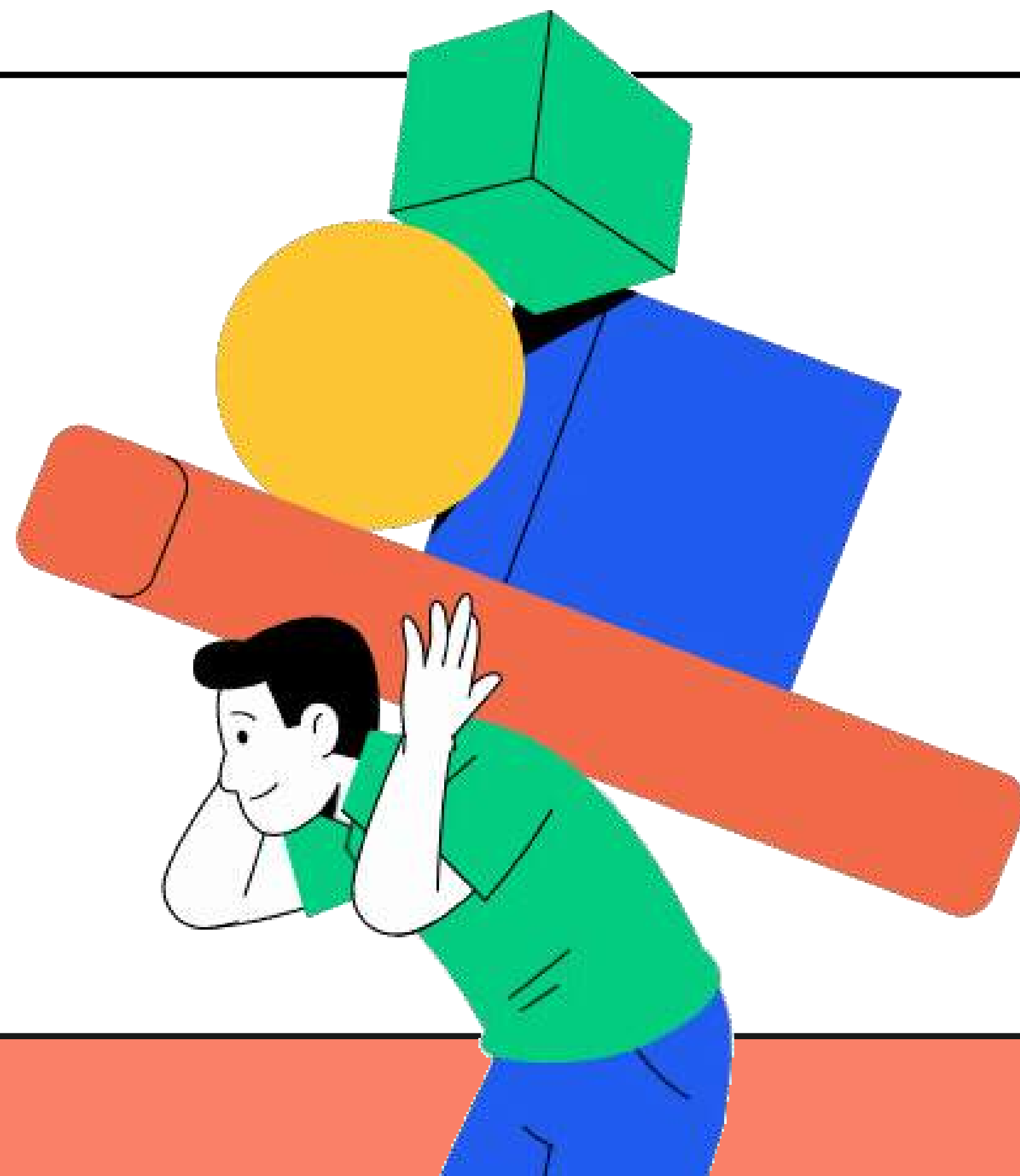
BUSINESS OBJECTIVE

Customer churn is a concerning problem for large companies (especially in the Telecom field) due to its direct effect on the revenues. Companies often seek to know which customers are likely to churn in the recent future so that a timely action could be taken to prevent it.



PROBLEM STATEMENT

We have built a Logistic Regression Machine Learning model for a Telecom company to predict which customers are likely to churn (stop using their service in the future).



ANALYZING THE DATASET

DATA DESCRIPTION



The dataset provided for this activity consists of 11 features where 10 are independent features and 1 is a target variable. Features in this dataset are described as below:

Churn: 1 if customer cancelled service, 0 if not (Target)

AccountWeeks: number of weeks customer has had active account

ContractRenewal: 1 if customer recently renewed contract, 0 if not

DataPlan: 1 if customer has data plan, 0 if not

DataUsage: gigabytes of monthly data usage

CustServCalls: number of calls into customer service

DayMins: average daytime minutes per month

DayCalls: average number of daytime calls

MonthlyCharge: average monthly bill

OverageFee: largest overage fee in last 12 months

RoamMins: average number of roaming minutes

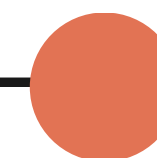
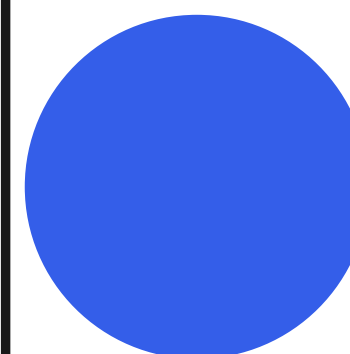
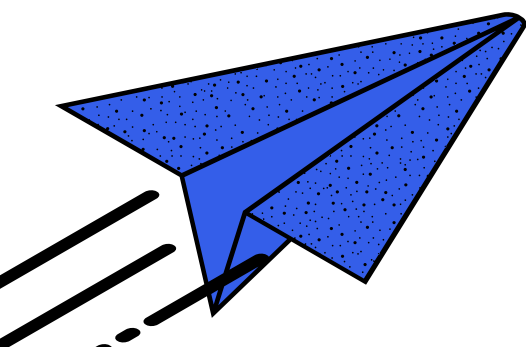
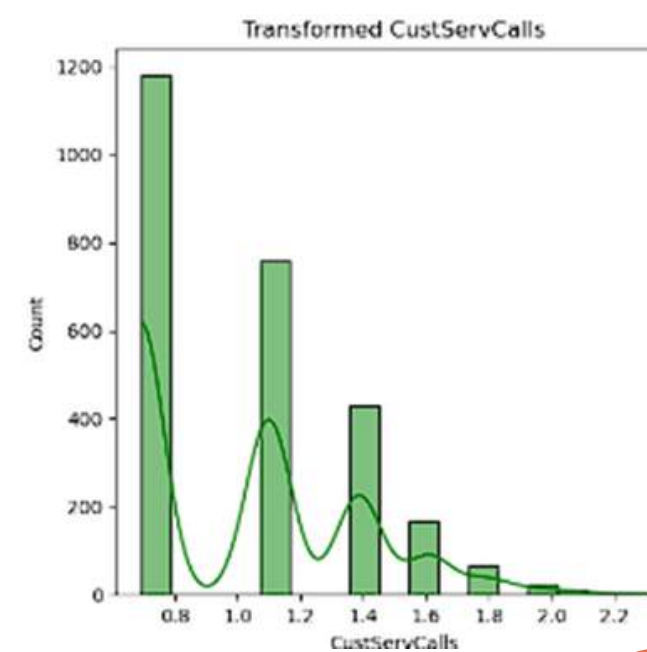
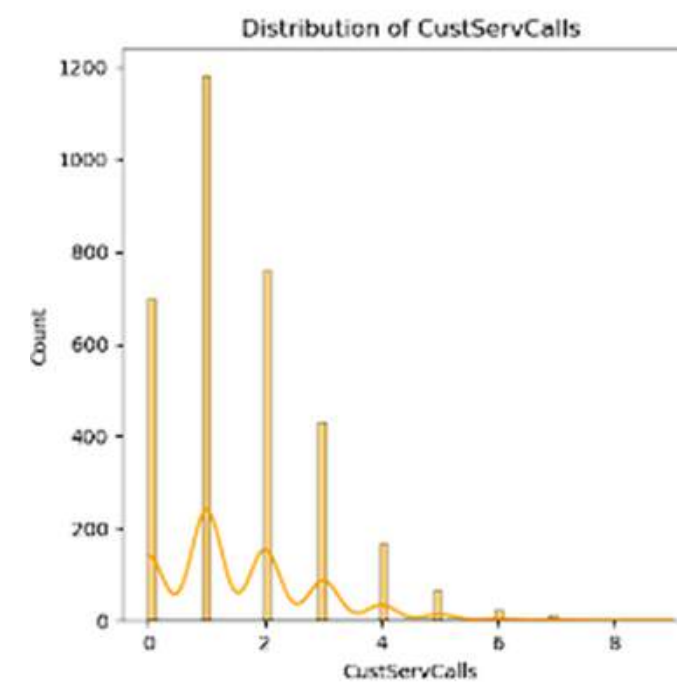
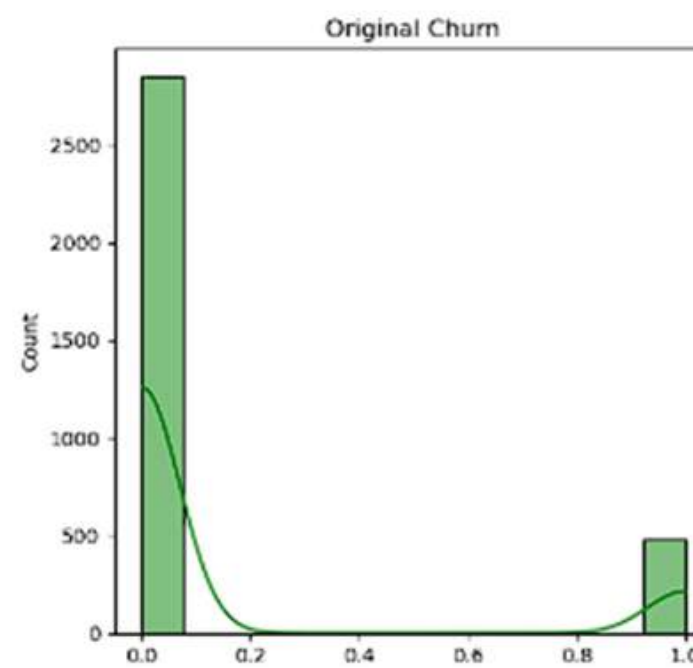
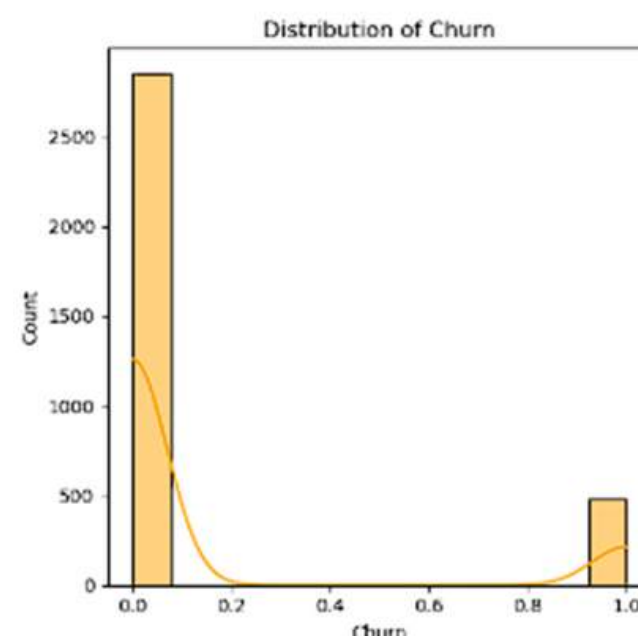
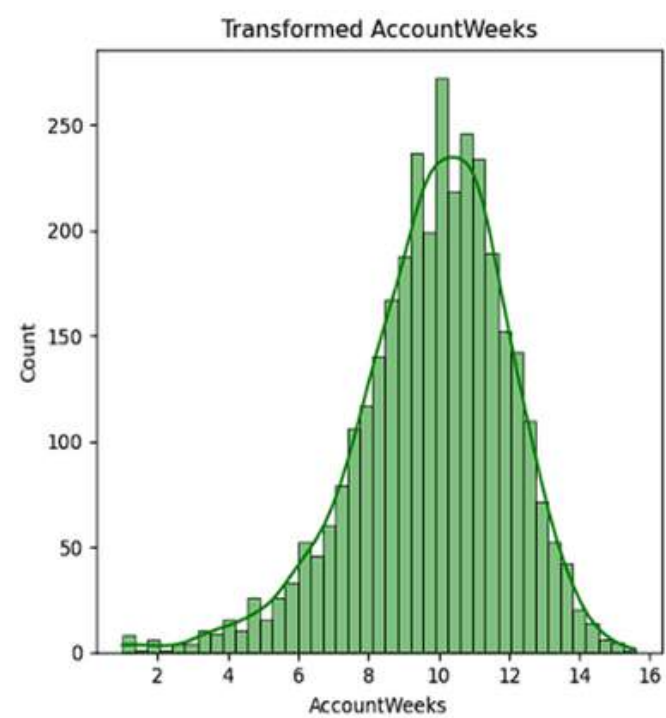
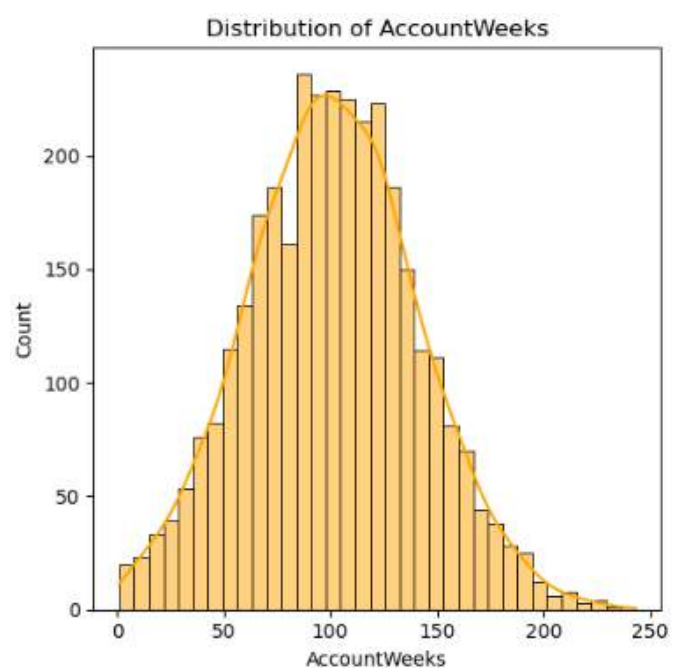


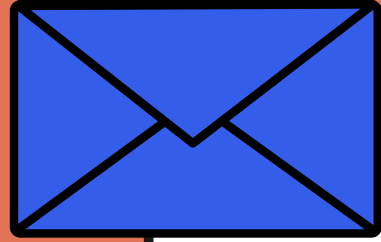


FEATURE TRANSFORMATION



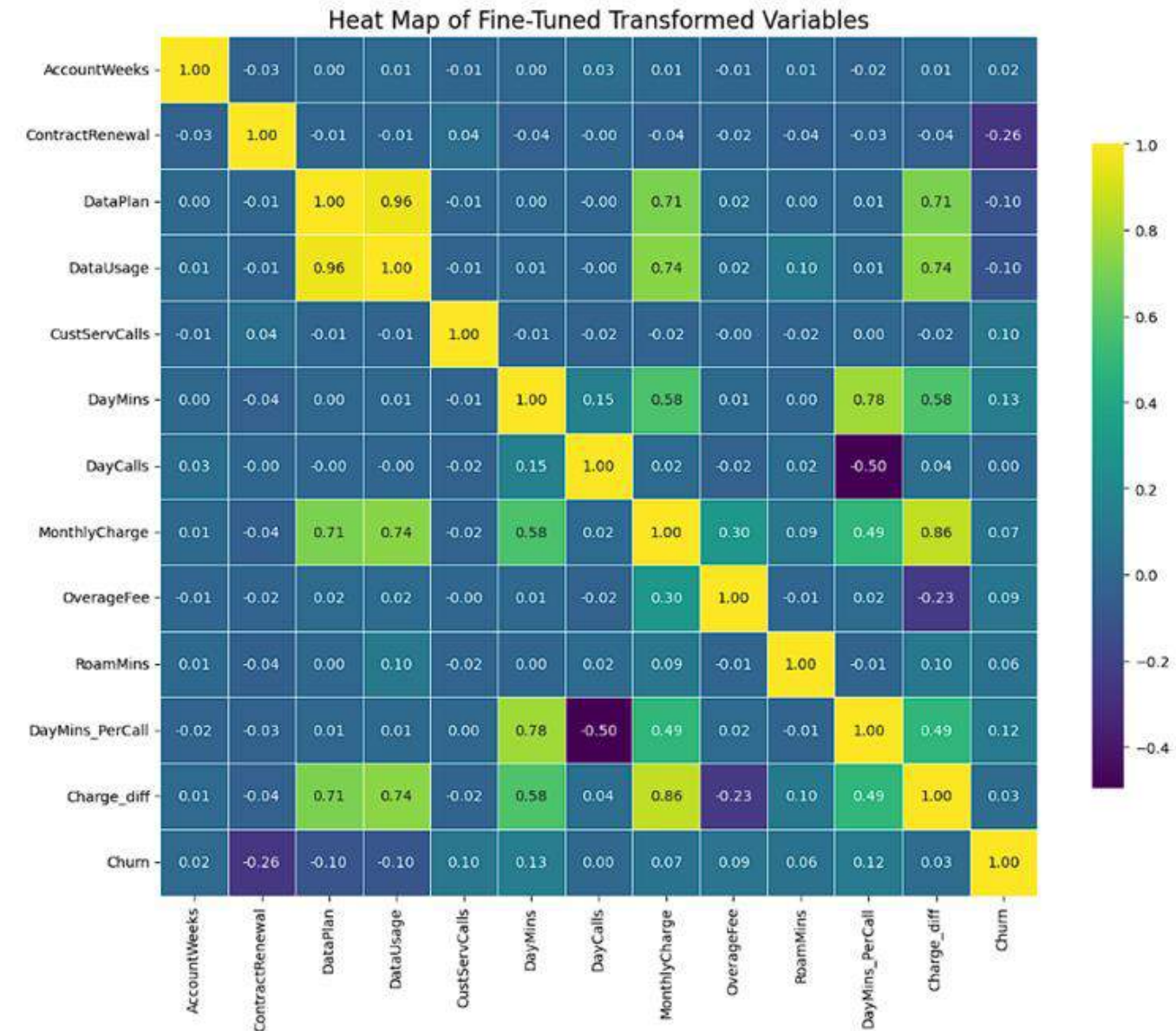
- Log transformation (`np.log1p`): Applied to AccountWeeks, CustServCalls, MonthlyCharge, and OverageFee.
- Square root transformation (`np.sqrt`): Applied to DataUsage, DayMins, DayCalls, and RoamMins.





HEAT MAP FOR TUNED TRANSFORMED VARIABLES

- Churn is directionally proportional to charge diff & Day mins.
- Charge diff is directly proportional to Day mins per call, Data usage, Data plan & Monthly charge.
- DayMins per call is directly proportional to charge diff, Day mins per call.

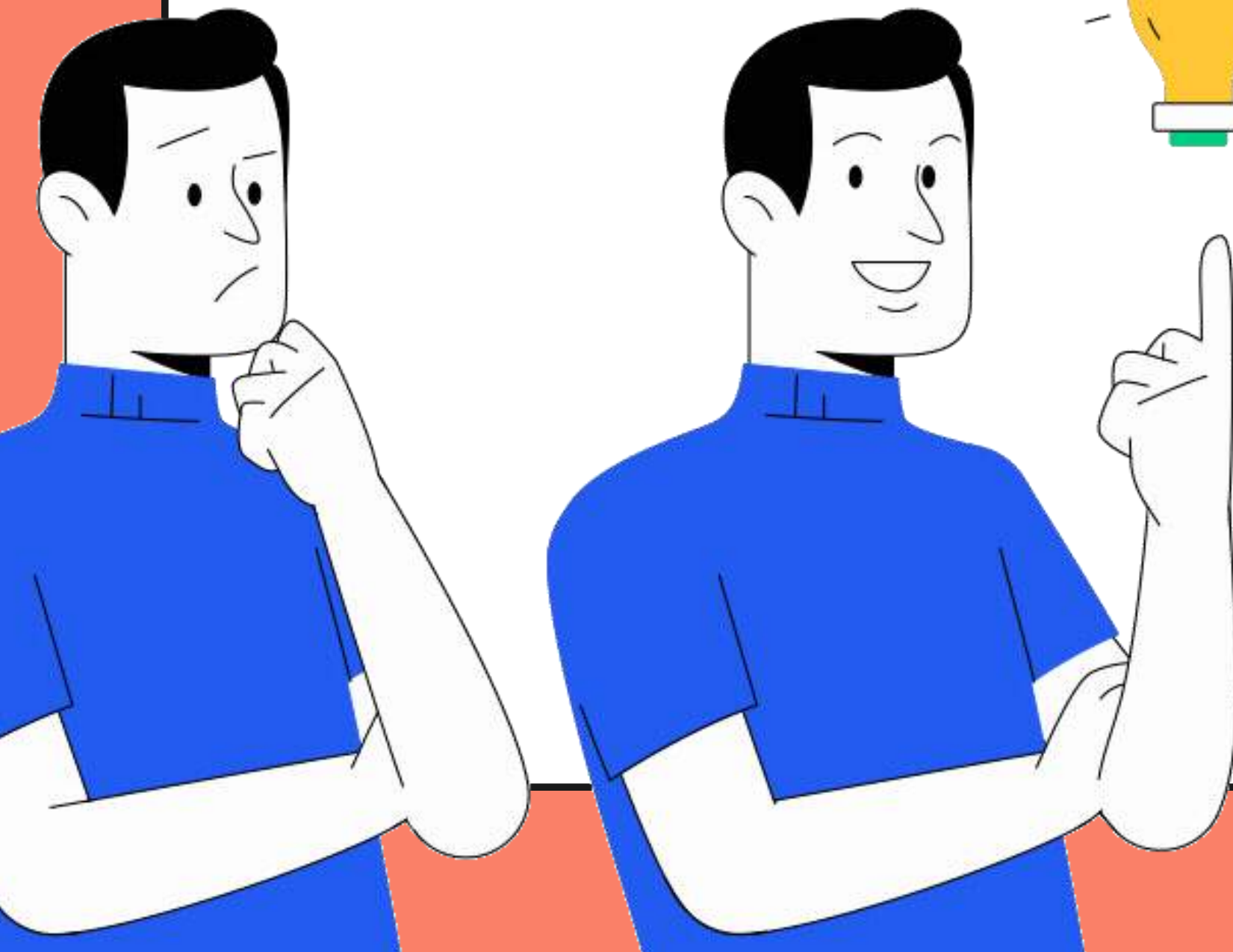


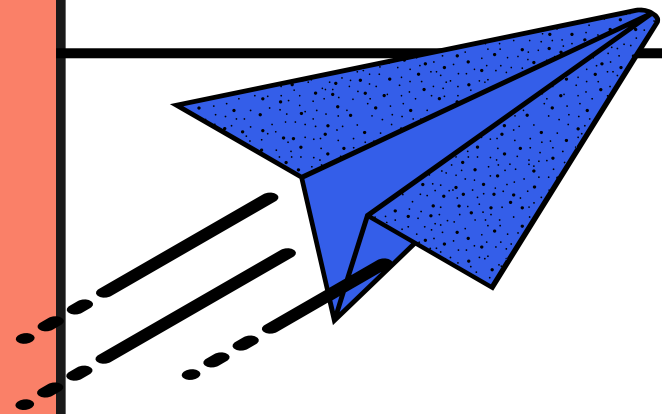


LABEL ENCODING & SMOTE

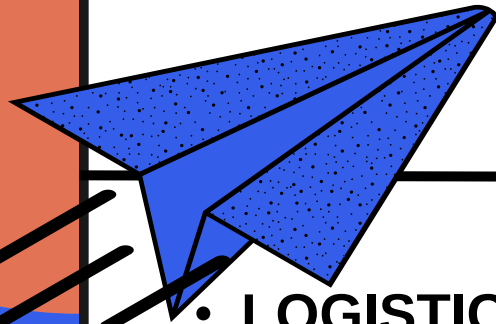


- We will convert the categorical variable "MonthlyUsageCategory" to numerical before applying SMOTE.
- Label Encoding is a technique used to convert categorical data into numerical format, which can be used by machine learning algorithms. In label encoding, each unique category value is assigned a numeric label.
- SMOTE (Synthetic Minority Over-sampling Technique) is a popular method used to address the issue of imbalanced datasets in machine learning. It works by generating synthetic samples for the minority class to balance the dataset.

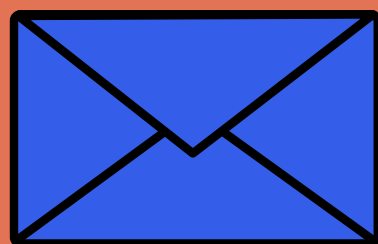




BUILD & EVALUATE MODELS



- **LOGISTIC REGRESSION:-**Logistic Regression is a linear model used for binary classification tasks. It estimates the probability that a given input belongs to a certain class using a logistic function.
- **DECISION TREE:-**A Decision Tree is a non-linear model that splits the data into subsets based on feature values, creating a tree-like structure of decisions.
- **RANDOM FOREST:-**Random Forest is an ensemble model that uses multiple decision trees to improve prediction accuracy and control over-fitting.
- **GRADIENT BOOSTING:-**Gradient Boosting builds an ensemble of trees sequentially, where each tree corrects errors from the previous ones.
- **K-Nearest Neighbours (KNN):-**KNN is a non-parametric algorithm that classifies instances based on the majority class among the nearest neighbors.
- **AdaBoost:-**AdaBoost is an ensemble method that adjusts the weights of incorrectly classified instances, boosting the performance of weak classifiers.
- **XGBoost:-**XGBoost is an efficient and scalable implementation of gradient boosting, often achieving high performance with its regularization techniques.
- **LightGBM:-**LightGBM is a gradient boosting framework that uses tree-based learning algorithms, optimized for speed and efficiency.

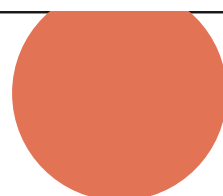
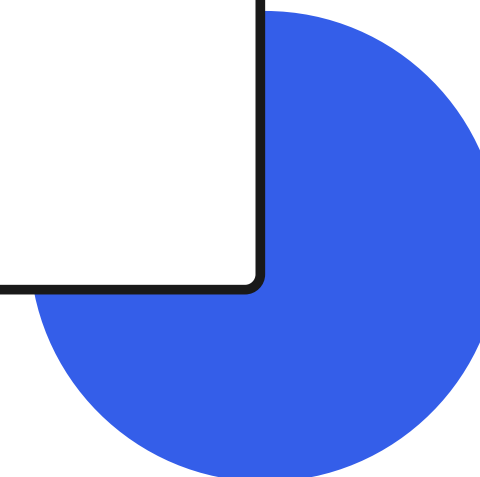


Accuracy score for Decision Tree, Random Forest & XGBoost is one.

F1-score for Decision Tree, Random Forest & XGBoost is one.

Precision score for Decision Tree, Random Forest & XGBoost is one.

	Metric	Logistic Regression	Decision Tree	Random Forest	Gradient Boosting	K-Nearest Neighbors	Gaussian Naive Bayes	Support Vector Machine	Multi-Layer Perceptron	AdaBoost	XGBoost	LightGBM	CatBoost
0	Accuracy	0.76	1.00	1.00	0.94	0.93	0.73	0.87	0.92	0.88	1.00	0.99	0.98
1	F1-score	0.74	1.00	1.00	0.94	0.94	0.70	0.87	0.92	0.88	1.00	0.99	0.98
2	Precision	0.80	1.00	1.00	0.94	0.89	0.78	0.88	0.91	0.90	1.00	0.99	0.99
3	Recall	0.68	1.00	1.00	0.93	0.99	0.64	0.86	0.92	0.86	1.00	0.99	0.97
4	AUC-ROC	0.76	1.00	1.00	0.94	0.93	0.73	0.87	0.92	0.88	1.00	0.99	0.98

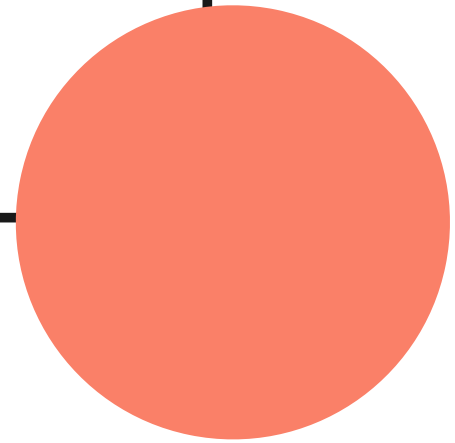




HYPER PARAMETER TUNING USING GRID SEARCH

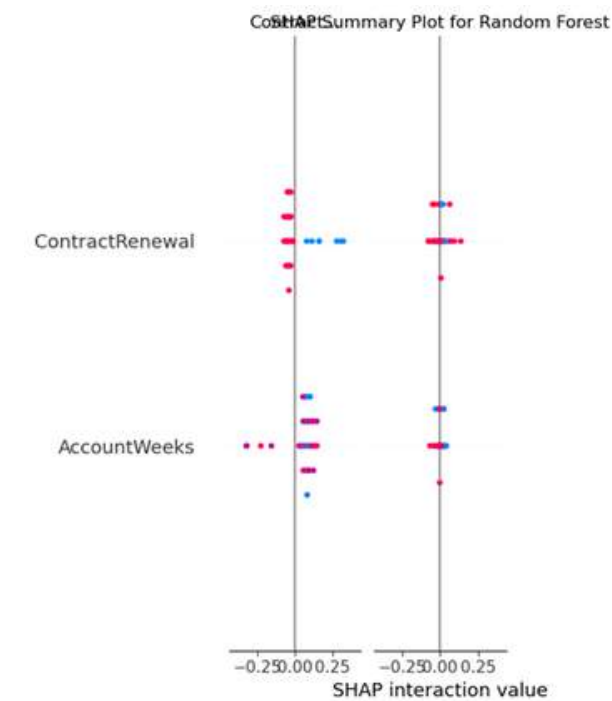
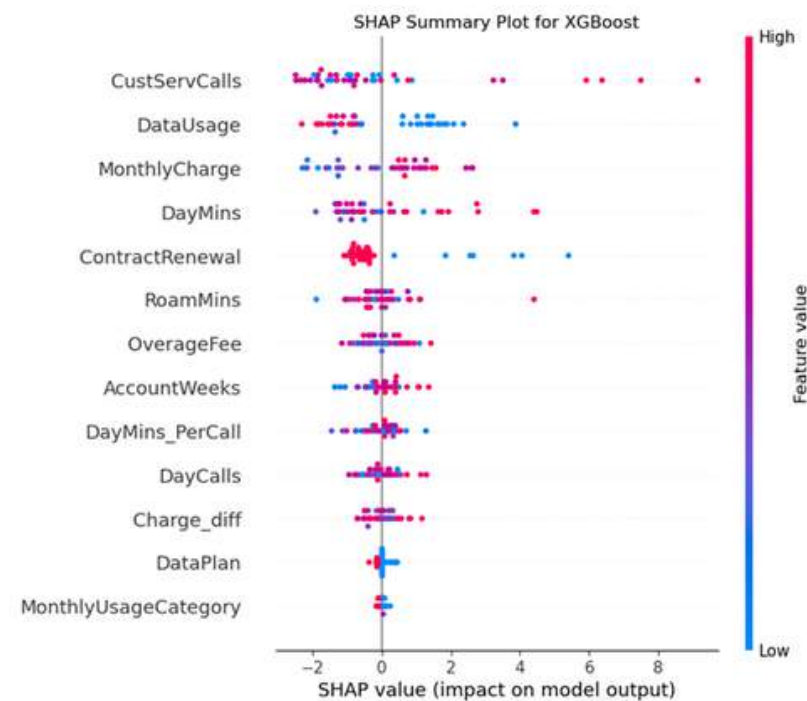
Hyperparameter tuning optimizes the performance of each model by searching for the best combination of parameters using GridSearchCV. This process ensures each model (e.g. Logistic Regression, Decision Tree, Random Forest, etc.) is configured with the most effective parameters to maximize accuracy.

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
Fitting 5 folds for each of 9 candidates, totalling 45 fits
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Fitting 5 folds for each of 18 candidates, totalling 90 fits
[LightGBM] [Info] Number of positive: 2850, number of negative: 2850
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.001038 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 2564
[LightGBM] [Info] Number of data points in the train set: 5700, number of used features: 13
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Fitting 5 folds for each of 16 candidates, totalling 80 fits
```



SHAP(SHAPELY ADDITIVE EXPLANATIONS)

SHAP explains the output of machine learning models by computing the contribution of each feature to the predictions. Using Tree Explainer and Kernel Explainer, SHAP values are computed to understand feature importance and visualize how different features impact model predictions

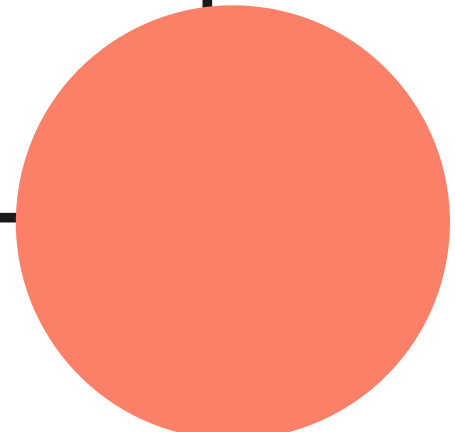




CROSS-VALIDATION

Cross-validation evaluates the generalizability of the hyperparameter-tuned models. Using Stratified Fold, the dataset is split into multiple folds, and each model is trained and tested on these folds to compute the mean accuracy and standard deviation. This process ensures the models perform consistently and reliably across different subsets of the data.

```
Logistic Regression: Mean Accuracy = 0.7588, Standard Deviation = 0.0103
Decision Tree: Mean Accuracy = 0.9040, Standard Deviation = 0.0084
Random Forest: Mean Accuracy = 0.9372, Standard Deviation = 0.0081
Gradient Boosting: Mean Accuracy = 0.9511, Standard Deviation = 0.0073
AdaBoost: Mean Accuracy = 0.8904, Standard Deviation = 0.0062
XGBoost: Mean Accuracy = 0.9519, Standard Deviation = 0.0043
LightGBM: Mean Accuracy = 0.9493, Standard Deviation = 0.0091
CatBoost: Mean Accuracy = 0.9554, Standard Deviation = 0.0081
KNN: Mean Accuracy = 0.9058, Standard Deviation = 0.0030
SVM: Mean Accuracy = 0.8879, Standard Deviation = 0.0057
MLP: Mean Accuracy = 0.9195, Standard Deviation = 0.0054
```



DEPLOYEMENT OF MODEL



Deploying a machine learning model involves several steps, from developing the model to deploying it so it can be used in production. While Spyder is an IDE primarily for development and analysis, you can still prepare your model in Spyder before deploying it. Here's a general workflow for deploying a machine learning model, using Spyder for development.

Step 1:

Model Development in Spyder

- Data Preparation
- Model Training
- Model Evaluation
- Model Serialization

1

Step 2:

Prepare for Deployment

- Choose Deployment Method
- Create a Deployment Script
- Testing the API:
- Deploy to a Server

2

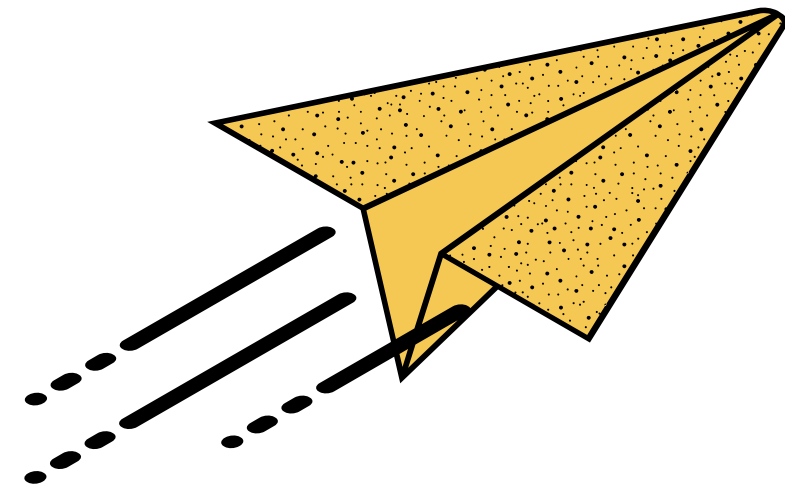
Step 3:

Monitor and Maintain

- Logging and Monitoring:
- Regular Updates:
- Scaling

3





**THANK
YOU!**

