<u>Project Introduction / Motivation</u>

In this project I worked with a fictional Telecom company's dataset made by IBM consisting of 7043 rows and 21 features. I was motivated to work with this dataset as it is applicable to many other businesses that have revenue streams coming from subscriptions. The telecom company's outcome variable is Churn resulting in yes or no. Yes in churn means the specific customer has decided to end services with the company, no meaning they continued subscribing to the services with the telecom company. My goal in this project is to be able to find a model that has the best performance metrics in classifying whether or not a customer is likely to Churn or not Churn. My hypothesis is that any classification model should be able to get ample results as the outcome variable is a binary classification problem.

<u>Exploratory Data Analysis</u>

One thing to highlight is that the majority class is people that decided to not churn at 73% while people that churned are 27%. This means it could present the problem of class imbalance and we would have to resample the outcome variable.

After iterating through each feature and finding the distribution on the churners, we can derive some conclusions through the visualizations.

Evenly distributed

Gender, PhoneService, MultipleLines are evenly distributed in the ratio between churners and non-churners. (these features could be insignificant)

Churn likely

Being a SeniorCitizen is more likely a churner, having no Partners and no Dependents are likely Churners. A customer having a month to month contract, having paperless billing, and having electronic checks are likely to churn.

Non-Churn likely

Having FiberOptic service and having no OnlineSecurity are likely Churners,
Having no OnlineBackup, no DeviceProtection, no TechSupport, no InternetService, no StreamingMovies are likely to churn.

Modeling Baseline

In my baseline modeling I took the cleaned data that is rid of all null rows and sent it straight into modeling with any feature selection (other than dropping CustomerID) or normalization. I used various classifiers: Random Forest, Knn, Logistic Regression, XGB, and LGBM. my def model() function provides me the cross validation score and ROC_AUC score for each classifier and my model_evaluation() function provides me the confusion matrix for each classifier.

The 2 best performing models are the XGB and LGBM:

	XGB	LGBM
Cross Val Score	84.78%	84.72%
ROC_AUC Score	71.37%	71.49%

The above 2 best performers are all tree based algorithm.

The worst performing model is Knn:

	Knn
Cross Val Score	77.87%
ROC_AUC Score	67.63%

From this I derived that tree based algorithms most likely perform best for this binary classification problem.

After UpSampling the outcome variable to be evenly distributed, the 2 best are still XGB and LGBM:

	XGB (ReSampled)	LGBM (ReSampled)
Cross Val Score	86.78% (+2%)	86.79% (+2.07%)
ROC_AUC Score	75.46% (+4.09%)	76.18% (+4.69%)

After resampling the outcome variable to an evenly distributed manner, we see a strong performance increase.

Modeling with modifications on dataset
After scaling data using the normalization method with MinMaxScaler() in sklearn, and dropping the features that have < 0.1 correlation coefficient to the outcome variable [MultipleLines, PhoneService, gender, StreamingTV, StreamingMovies, InternetService]

	XGB	LGBM
Cross Val Score	84.61%	84.51%
ROC_AUC Score	72.22%	71.55%

After ReSampling outcome variable:

	XGB	LGBM
Cross Val Score	86.57% (+1.96)	86.60% (+2.09)
ROC_AUC Score	76.96% (+4.74)	77.22% (+5.67%)

It is very clear that resampling the dataset on the outcome variable boosts performance.

Modifications vs Baseline

The 2 best models performed slightly better on the modified dataset vs the baseline dataset (without resampling and with resampling on the outcome variable) One thing to note though is that the cross validation score drops slightly when comparing between baseline and modified dataset, but this slight drop is compensated with increase in AUC score, which is a better metric to determine how well the model reads the dataset. The conclusion is that the best performing model is LGBMClassifier.

Problems faced in project:

The problem I face in this project that is prominent across all models is the low precision and f1 score in the minority class (churners). My best model LGBMClassifier has a confusion matrix below:

	precision	recall	F-1 score
0 (non-Churners)	0.92	0.73	0.81
1 (Churners)	0.52	0.82	0.64

The only problem I am facing and could not solve is the low precision on the minority class, which also contributes to the low f-1 score. This means that when it comes to predicting churners, the model is great at detecting positive samples, but has a problem of over-predicting meaning that the model predicts positives that result in false positives. This problem could be from insufficient data in the minority class. In the modified dataset without resampling yet, the confusion matrix for LGBM is below:

	precision	recall	F-1 score
0 (non-Churners)	0.84	0.89	0.87
1 (Churners)	0.65	0.54	0.59

The problem after resampling seems to be sacrificing accuracy on non-churners in order to improve productivity in minority class. But within the minority class the problem is that we are improving the recall at the expense of precision. So yes after resampling we can catch a lot more positives but with the problem being having too many false positives!

Future Work:
In the future I hope to be able to address the problem stated above and find out the different ways possible to address this recall and precision problem in the minority class. Having tried oversampling, undersampling using sklearn's resampling and SMOTE, I am open to researching and finding out more ways to address this problem.

Citations:
https://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html
https://www.kaggle.com/datasets/blastchar/telco-customer-churn