Problem statement

To accurately classify customer churn for each client by leveraging tree-based models such as random forest and Extreme Gradient Boosting (XGB).

Dataset used: Link

EDA and Data Pre-processing steps taken

- ✓ Our dataset has 100,000 rows and one target variable in the form of a binary data type (0's and 1's).
- ✓ The following screenshot sums up all the available values in each column.

#	Column	Non-Null Count	Dtype
---	------	--------------	-----
0	rev_Mean	99643 non-null	float64
1	mou_Mean	99643 non-null	float64
2	totmrc_Mean	99643 non-null	float64
3	da_Mean	99643 non-null	float64
4	ovrmou_Mean	99643 non-null	float64
5	ovrrev_Mean	99643 non-null	float64
6	vceovr_Mean	99643 non-null	float64
7	datovr_Mean	99643 non-null	float64
8	roam_Mean	99643 non-null	float64
9	change_mou	99109 non-null	float64
10	change_rev	99109 non-null	float64
11	drop_vce_Mean	100000 non-null	float64
12	drop_dat_Mean	100000 non-null	float64
13	blck_vce_Mean	100000 non-null	float64
14	blck_dat_Mean	100000 non-null	float64
15	unan_vce_Mean	100000 non-null	float64
16	unan_dat_Mean	100000 non-null	float64
17	plcd_vce_Mean	100000 non-null	float64
18	plcd_dat_Mean	100000 non-null	float64
19	recv_vce_Mean	100000 non-null	float64
20	recv_sms_Mean	100000 non-null	float64
21	comp_vce_Mean	100000 non-null	float64
22	comp_dat_Mean	100000 non-null	float64
23	custcare_Mean	100000 non-null	float64
24	ccrndmou_Mean	100000 non-null	float64
25	cc_mou_Mean	100000 non-null	float64
26	inonemin_Mean	100000 non-null	float64
27	threeway_Mean	100000 non-null	float64
28	mou_cvce_Mean	100000 non-null	float64
29	mou_cdat_Mean	100000 non-null	float64
30	mou_rvce_Mean	100000 non-null	float64
31	owylis_vce_Mean	100000 non-null	float64
32	mouowylisv_Mean	100000 non-null	float64
33	iwylis_vce_Mean	100000 non-null	float64
34	mouiwylisv_Mean	100000 non-null	float64
35	peak_vce_Mean	100000 non-null	float64
36	peak_dat_Mean	100000 non-null	float64
37	mou_peav_Mean	100000 non-null	float64
38	mou_pead_Mean	100000 non-null	float64
39	opk_vce_Mean	100000 non-null	float64
40	opk_dat_Mean	100000 non-null	float64
41	mou_opkv_Mean	100000 non-null	float64
42	mou_opkd_Mean	100000 non-null	float64

42	mou_opkd_Mean	100000 non-null	float64
43	drop_blk_Mean	100000 non-null	float64
44	attempt_Mean	100000 non-null	float64
45	complete_Mean	100000 non-null	float64
46	callfwdv_Mean	100000 non-null	float64
47	callwait_Mean	100000 non-null	float64
48	churn	100000 non-null	int64
49	months	100000 non-null	int64
50	uniqsubs	100000 non-null	int64
51	actvsubs	100000 non-null	int64
52	new_cell	100000 non-null	object
53	crclscod	100000 non-null	object
54	asl_flag	100000 non-null	object
55	totcalls	100000 non-null	int64
56	totmou	100000 non-null	float64
57	totrev	100000 non-null	float64
58	adjrev	100000 non-null	float64
59	adjmou	100000 non-null	float64
60	adjqty	100000 non-null	int64
61	avgrev	100000 non-null	float64
62	avgmou	100000 non-null	float64
63	avgqty	100000 non-null	float64
64	avg3mou	100000 non-null	int64
65	avg3qty	100000 non-null	int64
66	avg3rev	100000 non-null	int64
67	avg6mou	97161 non-null	float64
68	avg6qty	97161 non-null	float64
69	avg6rev	97161 non-null	float64
70	prizm_social_one	92612 non-null	object
71	area	99960 non-null	object
72	dualband	99999 non-null	object
73	refurb_new	99999 non-null	object
74	hnd_price	99153 non-null	float64
75	phones	99999 non-null	float64
76	models	99999 non-null	float64
77	hnd_webcap	89811 non-null	object
78	truck	98268 non-null	float64
79	rv	98268 non-null	float64
80	ownrent	66294 non-null	object
81	lor	69810 non-null	float64
82	dwlltype	68091 non-null	object
83	marital	98268 non-null	object
84	adults	76981 non-null	float64
85	infobase	77921 non-null	object
86	income	74564 non-null	float64

87	numbcars	50634 non-null	float64
88	HHstatin	62077 non-null	object
89	dwllsize	61692 non-null	object
90	forgntvl	98268 non-null	float64
91	ethnic	98268 non-null	object
92	kid0_2	98268 non-null	object
93	kid3_5	98268 non-null	object
94	kid6_10	98268 non-null	object
95	kid11_15	98268 non-null	object
96	kid16_17	98268 non-null	object
97	creditcd	98268 non-null	object
98	eqpdays	99999 non-null	float64
99	Customer_ID	100000 non-null	int64

- ✓ Columns that had more 30% missing values were dropped from the dataset.
- ✓ The distribution of churn is checked with the help of countplot to make sure classification results for target variable are balanced.
- ✓ Although it is not recommended to use KNN for large datasets, I wanted to personally test it out to see how KNN performs on large datasets. Missing numerical values were filled with KNN and mode was used to fill out all the categorical columns.
- ✓ Outliers were removed using the Interquartile range limits with a slightly more lenient range. (3 times IQR instead of the usual 1.25)
- ✓ Because the dataset is big, I tried different imputation methods to get the maximum out of leveraging machine learning techniques. Different versions of the dataset with different imputations were taken to try and get the best accuracy.
  1. In the first version (df1), all the numerical columns are subjected to PCA (principal component analysis). Since the dataset has high dimensionality (99 independent variables), considering PCA for reducing feature complexity was a viable choice. Features that retained 95% variance in data were converted to principal components (Decomposed columns). The end result was reduction of number of variables from 100 to 44 with 26 principal components.
  2. In the second version (final_df), all the numerical columns are checked for multi-collinearity. Models such as XGB deal with multi-collinearity pretty well but this step was carried to check for accuracies across different iterations of pre-processed data. Columns with VIF > 5 (Variance inflation factor – checks how much a predictor variable is correlated with all other predictor variables) are eliminated from the dataset. VIF is only calculated for top 30 highly correlated features.
  3. A third version that is unaffected by above two steps is also fed into the models. The top 30 highly correlated features are only selected in this case for prediction.

Model Training and Evaluation

→A train-test split of 80-20 is allotted and stratify is passed as a parameter to make sure classes are balanced in test and training dataset.

→To reduce computational cost and time, all the necessary attributes for our tree model are passed into param_dist. By this method, randomized search CV uses the combinations only passed into param_dist. These parameters are a common starting point but try to cover most bases (different combinations) with respect to achieving high accuracy.

→ The same steps are carried for our XGB model.

→ Classification reports are generated for each variation of our dataset.

→ The model that had only the top 30 highly correlated variables (No VIF elimination) had the highest accuracy with 62% among the Random forest models. This was the same case with the XGB model.

→ Confusion matrices are plotted for the best models to understand proportions of True negatives to True positives along with other combinations.

Dataset variation	Random forest accuracy (%)	XGB model accuracy (%)
df1 (PCA)	50.8	51.4
Final_df (VIF)	60.8	61
Top 30 corr. features	61.5	**62.4**

Parameters passed into param_dist

n_estimators → Decides the total number of decision trees that will be used to train the model.

Learning rate → Controlling the contribution of each tree to final outcome. Learning rate will be same for all trees.

Max_depth → Increase the number of levels or nodes in our trees to fish out for more complex insights in data.

Min_child_weight → Can be used to limit how many data points are in a child node before the final split. An input of 3 means that nodes will be split until each child node contains at least 3 data points.

Subsample → Each tree will be assigned random subsets of the data. For eg, each tree will be trained on a random 70% of the dataset, and the remaining 30% of the data will not be used for that particular tree.

Colsample_bytree →Similar to subsample, a certain percentage of the features can be chosen randomly for training each tree. This can help the model to generalize well with unseen data.

Alpha, lambda and gamma → Used for performing L1 Regularization, L2 Regularization and loss reduction. These regularization parameters are used to dynamically change coefficient weights to prevent overfitting whereas loss reduction is used to make sure that a split takes place only if minimal error (or loss) is achieved.

Challenges faced

✓ Model was consistently overfitting with data. Finding a fix for overfitting proved to be a tedious task.

✓ An highly accurate test score was difficult to achieve. Hyperparameter tuning did not turn out to be efficient.

✓ Different pre-processing steps had to be taken for the dataset to figure out model with best accuracy.

Key library methods in use →

- ✓ Scikit-learn: NearestNeighbors, PCA, RandomizedSearchCV, RandomForestClassifier
- ✓ StatsModels: variance_inflation_factor, add_constant
- ✓ xgboost: XGBClassifier