Exploratory Data Analytics

11 data points were found to be missing under the TotalCharges column. As the points are minimal and reasons were unknown it was removed to maintain consistency.

Through a heatmap, it is shown that customers with “No internet service” under the additional telco services columns and “No phone” under the Multiple Lines column could be converted to “No” as they are highly correlated to it and are essentially the same.

To determine the demographics of the customers and spot for any relation between churn rate and the features, a series of graphs (bar graphs/boxplot/violin plot/distribution graph) were utilised. In addition, pairplots and heatmaps are used to measure correlation between features and 3-dimensional scatter plots are used to do multivariate analysis. Generally, Customers with fiber optic, month-to-month contracts, paying through electronic checks with short term tenures and high monthly charges were shown to have a higher churn rate.

Data Pre-processing

Entries in binary categorical features were converted to 1 and 0 using labelencoder() while entries in multi valued categorical features were converted to dummy variables using get_dummies(). Lastly, minmaxscalar() is used to normalize all numerical features into a common scale. It is important to pre-process the data so that all features can be used for modelling and standardizing the features to a common scale ensures that equal weight is given to all features so that a fair and accurate model is built.

Feature Selection

As the dataset contains 19 features and not all are useful, feature selection is needed to maximise model performance and prevent overfitting on the train dataset. Using chi2 as the scoring function, each feature is evaluated based on its score and p-value. gender and PhoneService are removed as its p-value is less than 0.05 signifying its statistical insignificance in the model. TotalCharges is also dropped as it is highly correlated to tenure and MonthlyCharges and addition of it will not improve the final model.

Model Building

Accuracy and sensitivity are metrics used to rank the models. In a business perspective, it is desirable to have high sensitivity (Most churn customers identified) at the cost of precision since failing to recognise these customers leads to greater loss for the telco company.

Stratification is used during train_test_split() to reduce variability between the train and test set and to maintain data integrity of the original dataset to achieve a more representative and accurate model.

To counter the unbalance dataset, oversampling methods such as SMOTE (creating synthetic data points) or setting a balanced class weight is utilized. Both methods reduce overfitting and information leakage since it does not duplicate existing data points.

Models explored are those introduced in class (Decision tree, ensemble methods, logistic regression and KNN classifier) with an addition of support vector machine. Every model is built and then fine-tuned using GridSearchCV. Results of the models are in the appendix.

Performance of learning models varies on data as they make different assumptions and have different rates of convergence. It is important to test them out. In this case, the baseline trees and ensemble methods performed better in accuracy compared to Logistic Regression and SVM. While the Logistic Regression and SVM have higher sensitivity rate. The strength of each model is noted and fine-tuned.

Different learning model requires different constraints, weights or learning rates to generalize data patterns. These constraints are hyperparameters and could be tuned to optimize model performance. Utilising GridSearchCV, the Trees and Ensemble models are tuned on its criterion, max depth, max features, learning rate and n_estimator. Logistic Regression is tuned on its penalty, max iteration and C. KNN in finding its optimal n_neighbours, weights, leaf_size and SVM in finding the optimal kernel, class_weight, gamma and C. The top two models with the highest accuracy and sensitivity is the adaBoost and Logistic Regression. The final model to recommend will depend on the purpose of creating the model.

Conclusion

To improve the model, we can explore stacking (model ensembling) where multiple different models are combined to generate a new model. As it can highlight good models and discredit bad one, the resulting model outperforms the base models. Feature creation: Creating new features from existing one to introduce new information into the model. Adding more data such as customer's service rating, income bracket and a wider age range and whether customer is exposed to advertisement from rival companies. Neural networks which have proven to be highly accurate at classification could be explored too.

Appendix

Baseline Models

Model	Accuracy_score	Sensitivity	Precision	f1_score	Roc_Auc_score
Decision Tree(With all Features)	0.727488	0.481283	0.487365	0.484305	0.648970
Bootstrap Aggregating with SMOTE	0.726066	0.597148	0.487627	0.536859	0.684952
Random Forest	0.784360	0.497326	0.617257	0.550839	0.692821
AdaBoost with SMOTE	0.773460	0.725490	0.556772	0.630031	0.758161
Logistic Regression with SMOTE	0.758294	0.793226	0.530393	0.635714	0.769434
SVM	0.704265	0.853832	0.469148	0.605563	0.751965

Fine-Tuned Models using GridSearchCV

Model	Accuracy_score	Sensitivity	Precision	f1_score	Roc_Auc_score
Decision Tree(Feature + Depth Selection)	0.740284	0.787879	0.507463	0.617318	0.755463
Bootstrap Aggregating(GridSearchCV)	0.757346	0.668449	0.534950	0.594295	0.728995
Random Forest(GridSearchCV)	0.788152	0.522282	0.620763	0.567280	0.703362
AdaBoost(GridSearchCV)	0.783886	0.700535	0.577093	0.632850	0.757304
Logistic Regression(GridSearchCV)	0.758294	0.793226	0.530393	0.635714	0.769434
KNN(GridSearchCV)	0.783886	0.499109	0.615385	0.551181	0.693066
SVM(GridSearchCV)	0.663981	0.889483	0.435428	0.584651	0.735897

References

1. 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset | https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
2. Feature selection for Machine Learning in Python | https://machinelearningmastery.com/feature-selection-machine-learning-python/
3. Why accuracy alone is a bad measure for classification tasks, and what we can do about it | Tryolabs Blog | https://tryolabs.com/blog/2013/03/25/why-accuracy-alone-bad-measure-classification-tasks-and-what-we-can-do-about-it/
4. Why Stratify? | http://wiki.landscapetoolbox.org/doku.php/general_design_topics:why_stratify
5. Hyperparameter Tuning the Random Forest in Python | Towards Data Science |

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

6. Hyperparameter optimization | https://en.wikipedia.org/wiki/Hyperparameter_optimization
7. A Kaggler's Guide to Model Stacking in Practice | http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/
8. Feature Creation | https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781784396053/5/ch05lvl1sec30/feature-creation