Machine learning steps

1. Separate the test and training datasets before starting to explore the data more.
2. Perform EDA on the data. Use describe, head and then some plots to get a better understanding of the data.
3. Look at the correlation between variables.

For numerical variables you can use the following code

Corr = df.corr()

Corr[‘Target’].sort\_values(ascending = Fasle)

Close to 1 means strong positive correlation and close to -1 means strong negative correlation.

This code uses the pearson r test for correlation and is only a test for linear correlation.

Another way to see the correlation between numerical features is the scatter\_matrix() function, which plots every numerical attribute against every other numerical feature.

1. Experiment with attribute combinations. **(This could be part of the feature engineering section.)**

If the data has a tail heavy distribution it may be better to transform them to have a more bell shape.

Have a look at the data and see what combinations of the features would be interesting to make a new feature with. Then have a look at the correlation of the new feature with the target.

**Prepare the data for ML algorithms**

Instead of doing this step manually rather write functions to do this.

1. If it a good idea to make a copy of the training dataset before making any changes on the data. It may also be a good idea to separate the target from the other data since you do not want to perform transformations on this data. It will probably be better to do this after you have looked at duplicates and missing values so it is easier to re- join the data.
2. Cleaning the data. Start by looking at missing values and decide how to deal with them.

Scikit-Learn has a great class called SimpleImputer which can help take care of the missing values. The first step is to create a simpleImputer instance which states that you want to impute the missing values with median for example.

From sklearn.impute import SimpleImputer

Imputer = SimpleImputer(strategy=”median”).

**Note:**

You can only calculate the median on numerical features so you will need to create a dataset with only numeric features.

df\_num = df.drop(columns=[])

imputer.fit(df\_num)

It is nice to do this since there might be more missing values in the data when the model goes into production compared to the data used to create the model.

Now the ‘trained’ imputer can then be used on the training set by replacing the missing values with the learned medians

X = imputer.transform(df\_num)

Then to put it back into the main df

df\_train = pd.DataFrame(X, columns = df\_num.columns, index =df\_num.index)

**Note:**

Graphical user interface, text, application

Description automatically generated

This is code I have previously used.

**Text and categorical attributes**

1. If you have a categorical variable that is ordinal, for example bad, average, and good you use the sklearn ordinal encoder.

From sklearn.preprocessing import OrdinalEncoder

Ordinal\_encoder = OrdinalEncoder()

Look for examples of this online

1. If the categorical feature is not ordinal use the one hot encoder.

Graphical user interface, text, application, email

Description automatically generated

**Feature Scaling**

There are two common ways to get the attribute to have the same scale, they are min/max scaler and standardization.

The min/max scaler also called normalization ( it doesn’t change it to normally distributed) changes the scale of the data to lay between 0 and 1

The standardization changes the data to have a mean of 0 and standard deviation of 1. This method should be used if your data already follows a bell curve shape

**Note:**

You must fit the scaler to only the training data and not the test set. Once you have the scaler ‘trained’ the you can use them to transform the training set and the test set

**Transformation Pipeline.**

An example of a small pipeline for the numeric features might be the following

Graphical user interface, text, application

Description automatically generated

If you want to transform the numerical and categorical data in one pipeline there is a function called ColumnTransformer which allows you to do both in one pipeline.

Graphical user interface, text, application

Description automatically generated

Look for some examples on this

Findings from the EDA

**Comparing the features with churn**

Categorical

* There is and even split between the number of males and females in the data set. The amount of males and females which churn is basically the same .
* Most of the customers (84%) are not senior citizens. Out of the younger customers 24% churn where 42% of the older customers churn
* There is almost and even split between having a partner or not. 13% more without a partner will churn.
* 30% of the customers have dependents and 16% of these will churn where 31% without dependents will churn.
* From churn by contract type we can see that nearly half of the month-month customers will churn and this group makes up 55% of the customer which is quite large.

The one- and two-year contract types have a lower churn rate with one year have 11% and two year only 3%. This shows that the longer the contract type the lower the rate of churn is. (**important**)

* More customers that get a paper bill will churn compared to paperless billing.
* From churn but payment method one can see that nearly half of customers with electronic checks churn and this makes up 34% of the customers.
* From the internet service one sees that nearly half of the fiber optic customers churn and this makes up 44% of the customers.

Numeric

* A lot of customers have been with the company of only one month, also a lot of customers have been with the company for more than 70 months.

This makes sense because customers with the highest churn belong to contract type of month to month and many leave within the first month.

* With monthly charges, the higher the monthly charges the higher the churn rate.

Observation: this could be because if you are one a month to month contract the monthly charges are more than if you have longer contract this would match that more customers with month-to-month contracts churn.

* With the total charges, the larger the total charges the lower the churn rate. This makes sense because there is a lot of customers with a tenure rate of only one month.

Things to consider with feature engineering.

* The data imbalance if that will affect the model.
* If we want to reduce dimensionality consider the following:
* Remove Phone Service since the multiple lines feature includes no phone services which as the same churn %. Multiple lines also provides more info of customers that have phone lines by splitting it into multiple lines or not.
* change the no internet service from online security, online backup, device protection, tech support, streaming TV and Streaming Movies to no. This will reduce the dimensions once the values have been one hot encoded, I am not sure if this would make a difference.

Model findings using auto ML

I tried two different datasets with the Auto ML.

1. **only cleaned dataset without feature engineering.**

The best selected model was the Voting Ensemble

Graphical user interface, text, application, chat or text message

Description automatically generated

Chart, treemap chart

Description automatically generated

1. **Cleaned dataset with min/max scaler, one hot encoder and feature engineering.**

The feature engineering included removing phone service because it was correlated with multiple lines, combing streaming TV and streaming Movies into one. I also removed no internet service in some of the features into the subset of no. Maybe try test without doing this.

Graphical user interface, text, application, chat or text message

Description automatically generated

From this table we can see that both the accuracy and precision have improved by doing this.

Chart, treemap chart

Description automatically generated

From the confusion matrix in both of the autoML runs we can see that nearly 50% of the minority class (customers that churn) are predicted incorrectly.

This could be because the dataset is imbalanced. Maybe try use a sampling technique to balance out the dataset before modelling.

Considerations

* When building the model, maybe try and decrease the precision to increase the recall. It could be better to have more false positives then missing true positives. Especially when it comes to customer chur.

Customer input would be create for this to know what the trade off between them.