Solution Architecture:

-Low Level Design

Telecom Customer Churn

Written By	Devanshu Shrivastava
Document Version	1.0
Last Revised Date	02-Jan-2022

Document Control

Change Record:

Version	Date	Author	Comments
1.0	02-Jan-2022	Devanshu Shrivastava	Draft Version of Document.

Reviews:

Version	Date	Reviewer	Comments

Contents

1. Introduction

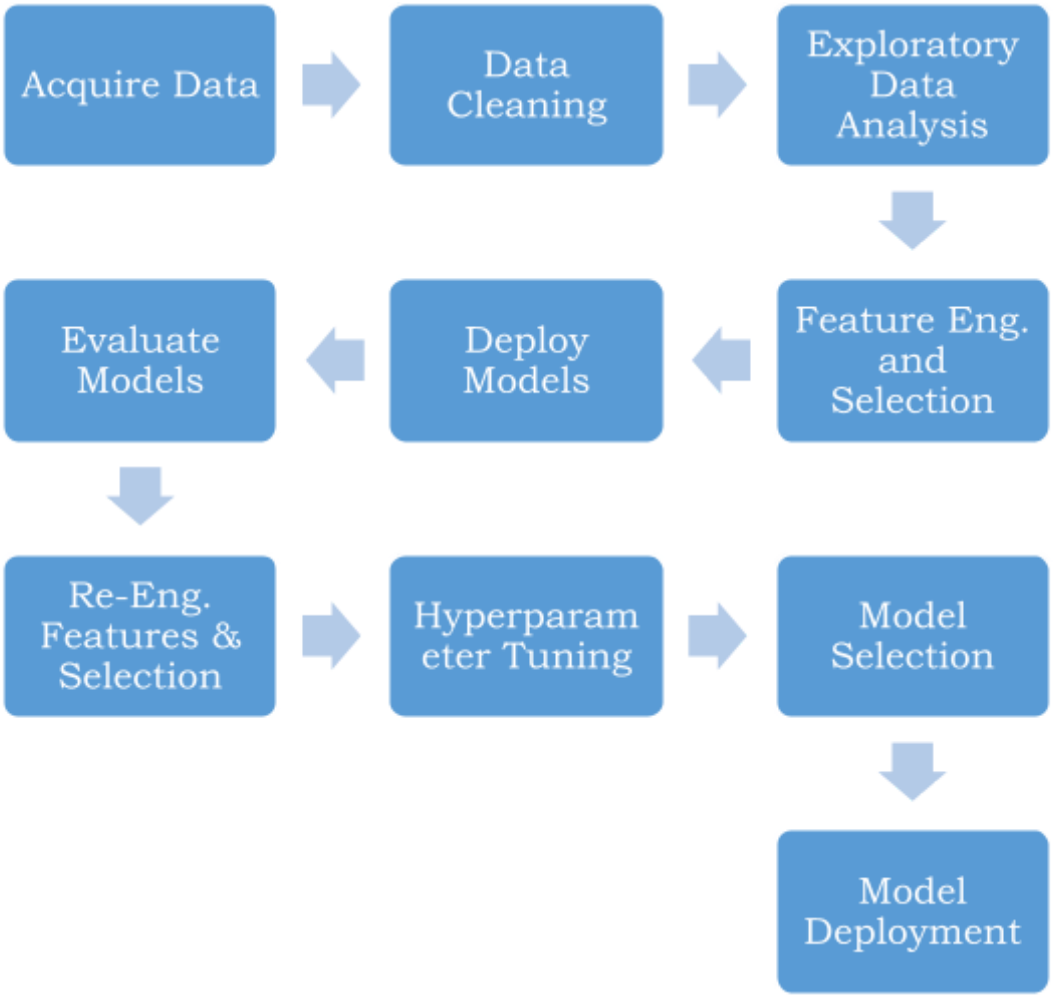
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual implementation of Telecom Customer Churn Use case. LLD describes the class diagrams with methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture



3. Architecture Description

3.1. Acquire Data

Provided dataset contains 7043 records for sample population data of telco customer acquired from the service desk as part of input to the Use Case that needs to be designed. Dataset contains 21 attributes out of which last attribute “Churn” is the target attribute. Each attribute is described as below.

Each row represents a customer, each column contains customer’s attributes described on the column Metadata.

The data set includes information about:

- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- Customer account information – how long they’ve been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

3.2. Data Transformation/Cleanup

In the Transformation Process, we will convert data type of few fields like Total Charges to float as it has been derived to be object by panda’s interpreter. Also, Total Charges column has few records with blank space we shall replace the same with NaN and drop these records further from our dataset before applying in model. This decision has been taken after experimenting with imputation of this records by recalculating values using formulae tenure * Monthly Charges which does not improve the outcome of Model. When these records are dropped Model provides better results.

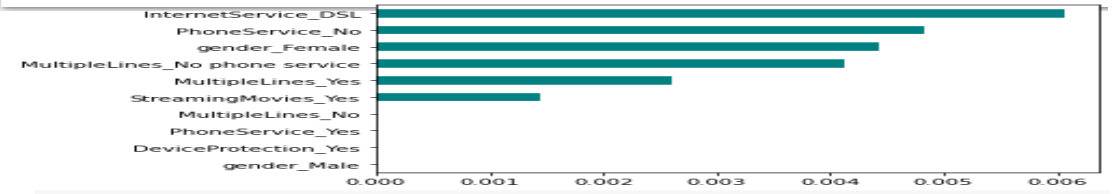
After removing the customer ID column as it provides unique identity to each record and thus denies variation in machine learning. It is observed that we have couple of duplicate records dropping this record was tested and it was observed it had no significant impact on model outcome thus data was kept as is and duplicates were not dropped.

3.3. Feature Engineering and Selection

This step includes applying methods to identify best n features which can be further used to improve model performance and weed out features which do not have significant impact or negatively impact the model performance. We have used methods like Variance Threshold, select K Best features, Information Gain and Correlation Heatmap to identify features with high variance or features which are highly correlated in which case only 1 of the correlated features will be used in the final model.

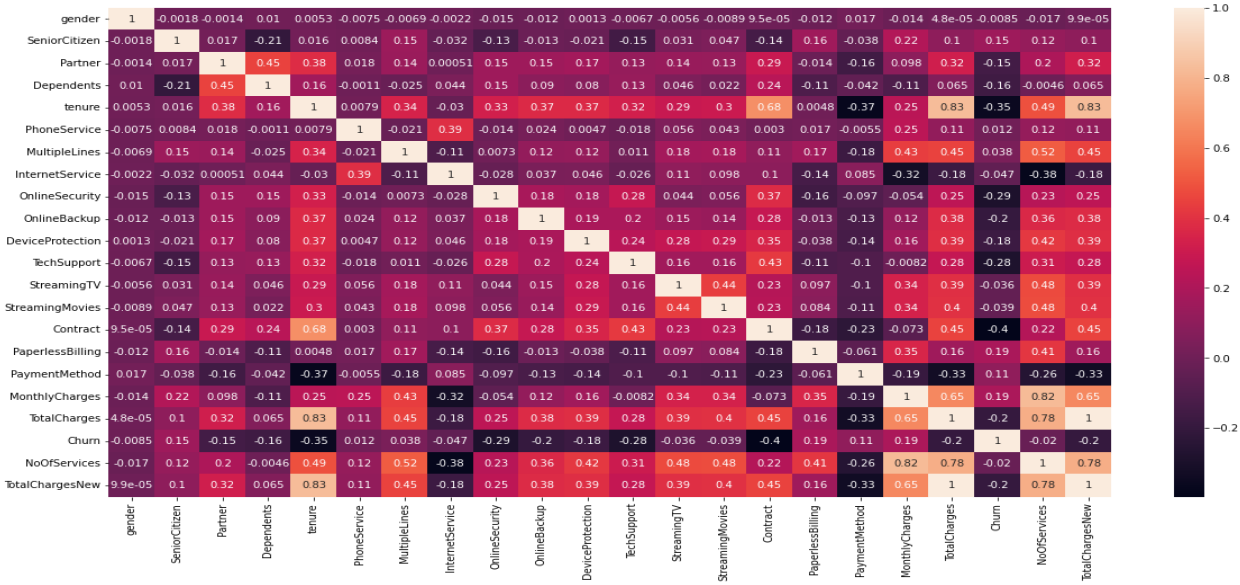
```
from sklearn.feature_selection import VarianceThreshold
selector = VarianceThreshold(1)
selector.fit(telcoDFXsc)
telcoDFXsc.columns[selector.get_support()]
```

```
Index(['Partner_No', 'Partner_Yes', 'Dependents_No', 'Dependents_Yes',
       'MultipleLines_No', 'InternetService_DSL',
       'InternetService_Fiber optic', 'OnlineSecurity_No',
       'StreamingMovies_Yes', 'Contract_One year',
       'PaymentMethod_Credit card (automatic)'],
      dtype='object')
```




```
from sklearn.feature_selection import SelectKBest, mutual_info_regression
#Select top 2 features based on mutual info regression
selector = SelectKBest(mutual_info_regression, k =10)
selector.fit(telcoDFX, telcoDFy)
telcoDFX.columns[selector.get_support()]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataCo
  y = column_or_1d(y, warn=True)
Index(['tenure', 'MonthlyCharges', 'TotalCharges',
       'InternetService_Fiber optic', 'OnlineSecurity_No', 'TechSupport_No',
       'StreamingTV_No internet service',
       'StreamingMovies_No internet service', 'Contract_Month-to-month',
       'Contract_Two year'],
      dtype='object')
```



3.10. Model Building and Evaluation

We Build and Evaluated 11 Classification Models with mentioned feature selections, data preprocessing, Standardization and Normalization mentioned in slides above. Best performing model is XG boost with accuracy of around 87.45 % while Linear Regression performed accuracy of 86.9 %.

Classification Algo	Accuracy	Precision	Recall	F1-score	Support
Logistic Regression	0.87	0.87	0.87	0.87	2582
Naïve Bayes	0.79	0.8	0.79	0.79	2582
Stochastic Gradient Descent	0.84	0.84	0.84	0.84	2582
KNN	0.85	0.85	0.85	0.85	2582
Random Forest	0.86	0.86	0.86	0.86	2582
SVC - Linear Kernel	0.86	0.86	0.86	0.86	2582
SVC - rbf Kernel	0.84	0.84	0.84	0.84	2582
Gradient Boosting	0.86	0.86	0.86	0.86	2582
AdaBoost	0.72	0.73	0.71	0.71	2582
XG Boost	0.87	0.87	0.87	0.87	2582
Cat Boost	0.86	0.87	0.86	0.86	2582

Applied RandomSearchCV to best performing model XG Boost after trying multiple iterations and hyper parameter combinations it was observed that the default values used initially provided the best results. Simlary RandomSearchCV was applied to 2nd best performing model Logistic Regression after trying multiple iterations and hyper parameter combinations it was observed that the default values used initially provided the best results.

Hyper parameter tuning XG Boost

```
[125] 1 from sklearn.model_selection import RandomizedSearchCV
      2
      3 params = { 'max_depth': [3, 5, 6, 10, 15, 20],
      4            'learning_rate': [0.01, 0.1, 0.2, 0.3],
      5            'subsample': np.arange(0.5, 1.0, 0.1),
      6            'colsample_bytree': np.arange(0.4, 1.0, 0.1),
      7            'colsample_bylevel': np.arange(0.4, 1.0, 0.1),
      8            'n_estimators': [100, 500, 1000]}
      9
     10 clf = RandomizedSearchCV(estimator=classifierXGB,
     11                           param_distributions=params,
     12                           scoring='neg_mean_squared_error',
     13                           n_iter=10,
     14                           verbose=1)
     15
     16 clf.fit(X_train, np.ravel(y_train))
     17 print("Best parameters:", clf.best_params_)
     18 print("Best score:", clf.best_score_)
     19 print("Best estimator:", clf.best_estimator_)
     20 print("Best index:", clf.best_index_)
     21 print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))

    Fitting 5 folds for each of 10 candidates, totalling 50 fits
    Best parameters: {'subsample': 0.6, 'n_estimators': 1000, 'max_depth': 10, 'learning_rate': 0.01, 'colsample_bytree': 0.4, 'colsample_by
    Best score: -0.14088309036587748
    Best estimator: XGBClassifier(colsample_bylevel=0.4, colsample_bytree=0.4, learning_rate=0.01,
                  max_depth=10, n_estimators=1000, subsample=0.6)
    Best index: 0
    Lowest RMSE:  0.37534396274068066
```

3.11. Deployment

We will be deploying the model to customer premise/Cloud based on where the customer requirement. Real time test data will be received from customer service desk via a Rest API which will be preprocessed, scaled/Normalized, and then passed to the model for prediction. Resultant outcome will be displayed on a simple dashboard or intimated to agents via push notification with general recommendations so action can be taken proactively so as to stop or reduce on Churn rate.

This is a workflow diagram for the Telco Customer Churn.