Introduction

Dataset is Telecom Churn Prediction Data. This has a set of mostly categorical features representing the services a customer subscribers to from the telecom company. There are 7043 data points in total

We need to add a time field as well. Have decided to add that based on the end of each week between June and August.

EDA

<<detail EDA can be found in Telco_Churn_EDA_Prediction.html and app\experiment\Telco_Churn_EDA_Prediction.ipynb >>

Some highlights →

- We don't have much missing/Null data. Only TotalCharges had missing values, only 11 records. So decided we can drop that data for now
- So now we have 7032 data points.
- Gender and Partner are more or less equally divided.
- Senior Citizen represent a smaller percentage of the sample
- Customers having Dependents are lower percentage of the sample
- month-month billing is the most common form of subscription
- While electronic check is the most popular method of payment, it is closely followed by the rest of the methods.
- There are not insignificant numbers of customers who do not have any internet service.
- Gender distribution among male and females are similar. It is not showing a clear difference between churn and retained.
- Customers over 65 years old are looking slightly more likely to churn
- Even though most customers have phone service, customer with or without service have a similar churn
- Type of contract seems to have interesting relationship with churn
- The distribution for Churn vs Not Churn is very different for Tenure and Monthly Charges.
- Using Cramer's V Correlation Analysis for categorical variables. We can see some huge correlation in couple of variables and some minor correlation between few others -
    - PhoneSerice and Multiple Lines seems to be highly correlated
    - Internet Service and online security, online protection, tech support and streaming seem to have above average correlation which is to be expected based on these services

Feature Selection

While from the EDA we can see that there are some significant features. Namely InternetService, Type of Contract, PaymentMethod, Tenure and Monthly Charges.

It is still a good idea to see the top best features from a RandomForest Model.
We can see after random forest model following feature seem to be the best ranked - InternetService, OnlineSecurity, TechSupport, Contract, PaymentMethod, Month, Tenure, MonthlyCharges, TotalCharges.

Based on this and our earlier EDA. Have decided to go with following set of features -

"InternetService", "month", "Contract", "PaymentMethod", "tenure", "MonthlyCharges"

Metric Selection

We have the task of predicting if a customer will churn or not. If we can successfully identify when a customer or a set of customers are about to churn we can run a campaign to target them to retain them. So for us accuracy matters the most. We are okay to have a few false positives as a result of this model as it would only be a minimal cost to run a campaign on few other customers. But it will help us retain most of the customers.

So the metric that we will be tracking will be accuracy

Model Evaluation

Decided to do evaluation between a series of models ->
Naive Bayes, kNN, Decision Trees, Random Forest

The idea is explore different models to see which has the best accuracy score and choose that for model training and predictions
(more details in the html and ipynb)

Model Selection

Based on the accuracy. Random Forest with 250 estimators and 15 as max depth is the best model and parameters that we should use

Model Training Application Dev and structure

The application dev part of this model will be under the following structure (all code included)

```
-data
        -- raw
        -- processed
- experiment
- models
- src
- evaluation
config
environment.yml
Makefile
```

Data

- This has the raw data, the processed data of adding the date field and the converted data of feature creation and conversion

Experiment

- This is the ipynb files that devs can use to experiment and iterate once the model is in production

Models

- This will store the actual pkl file that would be part of model deployment and application
- The pkl file will be used as an input to the model prediction process.

Src

- This has the source code for preparing dataset, creating features, training model and predictions using the model
- Idea of modularizing this is because at some point we might want to store all our features in a feature store
- Similarly for models, we might want to have a central repo/model monitoring service, eg - how many times a model has been updated. Audit on hyperparameter tuning etc

Evaluation

- This will have the evaluation metrics so that whenever the model needs to be updated we have the previous evaluation metrics and we know that is the baseline for this model
- It will be “metric” : “metric value”

config.toml

- This will include the model name, features, predictor variable and the model parameter.
- This will be used as the input to the model training process

Makefile

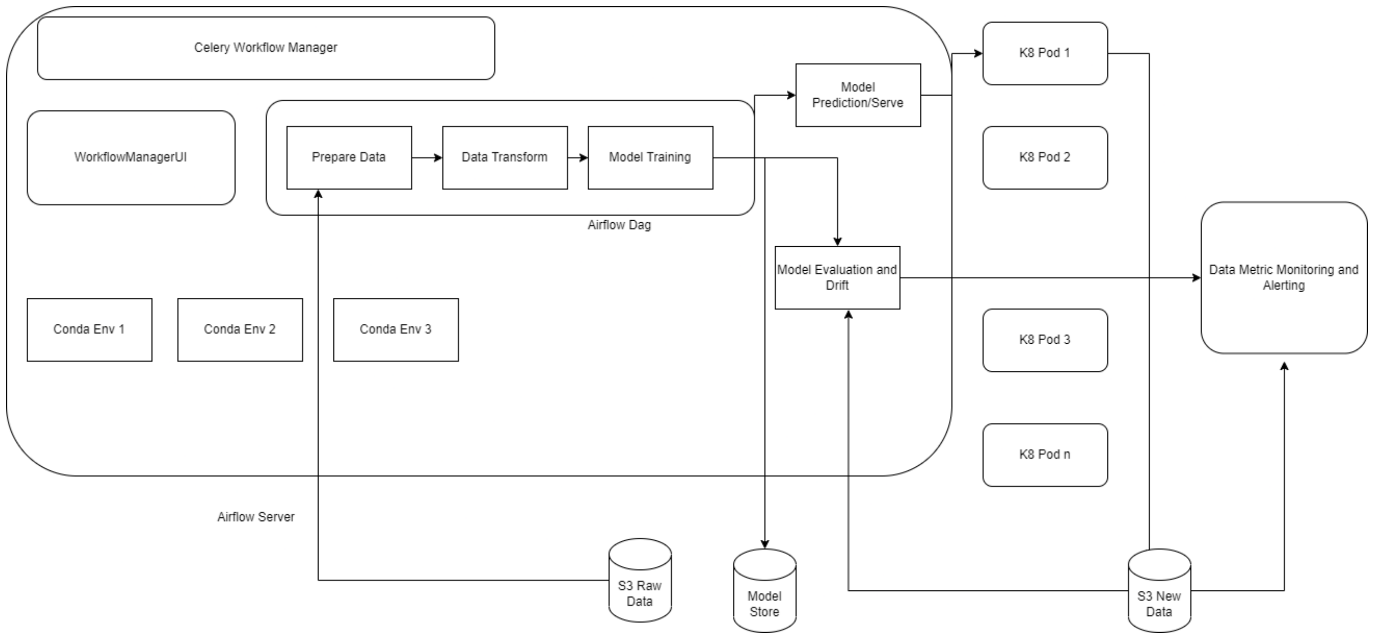
- Simple makefile file to have env setup, data preparation, feature creation, model training.

Running the scripts

Follow the instructions on readme.
Simple make commands for each step of the process

Model Deployment and Monitoring

If we are using airflow for batch based predictions
Each of the steps of the model will be part of DAG for the airflow



(high resolution picture attached as well)

1. Model Training will be part of a Airflow Dag
2. Airflow will act as the workflow manager/executor.
3. Each Step would be executed on a k8 cluster
4. After model training we can save the model in a data store
5. As new data comes in, if we are doing batch based predictions for churn we can run a daily/weekly application for customer identification.
6. As we get real data from earlier weeks. We can compare our predictions against the real values
7. We will also be monitoring the distribution of data. For example we know from our EDA. Tenure and monthly charges have a mean and median value. And both of these are used as features in our model. So significant difference in the new data on these values should trigger an alert
8. The model and data drift can be managed via an application that runs on real data and triggers alerts.