Machine Learning Engineer Nanodegree

Capstone Project Report

Title: Smart Inventory

Domain Background

Customer churn is a major concern for most of the telecom companies.

Customer churn is all about preferring other operators.Companies spend a lot in overcoming customer churn and companies adopt various techniques like they come up with special programs or incentives to targeted customers.But in case the target companies are predicted correctly company would be under loss.Hence its not a good idea to go with this approach rather use statistical techniques to predict customer churn.Some of the techniques could be neural nets,logistic regression etc

Check out the interesting article on customer churn from https://ieeexplore.ieee.org/document/8258400/

There are much more interesting articles related to “Business model and research on telecommunication operation transformation available at https://ieeexplore.ieee.org/document/5705218/?reload=true

Also check out https://data.gov.in where we have further research done in this area .you can obtain datasets related to Trends of Telecom subscribers in India as well as other areas e.g Telecommunication performance,Trends of Telecom subscribers in India, Internet and BroadBand subscribers ,teledensity in India etc

Problem Statement

This Capstone project is about enabling churn reduction using analytics and then implementing the most optimal Machine Learning model to reduce churn.

The objective is to predict customer churn. Since it’s a labelled data we can use we can use supervised learning and also the problem statement is to predict customer churn hence it’s a classification problem.

Datasets and Inputs

The datasets are taken from

- **https://www.kaggle.com/becksddf/churn-in-telecoms-dataset/data**
- **https://www.crowdanalytix.com/contests/why-customer-churn/**

The objective of this ML project is to predict customer churn. We are providing you a public dataset that has customer usage pattern and if the customer has churned or not. We expect you to develop an algorithm to predict the churn score based on usage pattern. The predictors provided are as follows:

account length:

total evening calls

total evening charge

international plan

total night minutes

total day minutes used

total night calls

voice mail plan

number of voice mail messages

number of customer service calls made

total night charge

day calls made

total international minutes used

total day charge

total international calls made

total evening minutes

total international charge

DataSet Information

The dataset describes mainly a how the customer churn is dependent on various factors e.g account length,total day calls,total evening calls,total day charge total night charge etc.

It has a total of 3333 observations.

Some of the factors like phoneno,areacode,state,international plan,voice mail plan may not be considered for predicting customer churn and hence shouldn't be considered

The target class which is customer churn is having a value of either true or false(0 or 1).

The following is the distribution of the customer churn of the 3333 observations provided:

```
Churn=Flase     2850
Churn =True      483
```

Which is around **16.9% of customer churn**.The idea is the idenfity the factors listed above w hich is impacting customer churn so that appropriate business actions are taken based on th ese factors

The sample of data is as shown below:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	cus s
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	2.70	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	3.70	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.78	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73	

5 rows × 21 columns

Benchmark Models

Since it's a classification problem,the following classification techniques can be used.

1.Logistic Regression

2.Random Forest

3.Ensemble Technique

Metrics

The following metrics can be considered for choosing the optimal classification model

1. F score

2.Training Time

3.Prediction Time

4.Cross Validation Results

Since its all about predicting whether the customer would churn or not , one metric to think about is accuracy score (it determine how accurately the customer churn can be predicted), but there is a imbalance in the data(churn v/s no churn) hence it wouldn't be good idea rather its always good to know the models ability to precisely predict the customer churn or recall of these customers the **F-score which is combination of precision or recall would be an important metric to consider**

The **Cross validation results** is all about how the model is performing on a train and test data with a train and test data split(there shouldn't be huge difference in the accuracy/f scores for both train and test data split)

The **training and prediction time** metrics would also be a deciding factor for model selection.

In case training and prediction time is given preference over accuracy we can compromise slightly on accuracy in decide on model selection based on the training and prediction time.

The metrics I would use to compare models would be **F score**.F score which is combination of both precision and recall would be a important metrics to consider.Accuracy can't be considered as the metrics to compare models as there is imbalance in data(churn v/s no churn)

F-beta score can be used as a metric that considers both precision and recall

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

In particular, when $\beta = 0.5$, more emphasis is placed on precision. This is called the F $_{0.5}$ score (or F-score for simplicity).

ANALYSIS

Data Exploration

The data set is a public dataset that has customer usage pattern and if the customer has churned or not. We expect you to develop an algorithm to predict the churn score based on usage pattern. The predictors provided are as follows:

account length:

total evening calls

total evening charge

international plan

total night minutes

total day minutes used

total night calls

voice mail plan

number of voice mail messages

number of customer service calls made

total night charge

day calls made

total international minutes used

total day charge

total international calls made

total evening minutes

total international charge

The sample of the data will look like:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	cus s
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	2.70	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	3.70	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.78	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73	

5 rows × 21 columns

Here is the brief description of the data:

The number of rows provided for prediction of customer churn is **3333**

Number of input attributes :**20**(E.g account length,International paln,phone number,total day minutes,total day charge etc)

Target Variable:churn(having value True/False)

DataTypes of variables:float,int,object,Boolean

The data distribution is:

```
Churn=Flase     2850
Churn =True      483
```

Which is around **14.49% of customer churn**.The idea is the idenfity the input attributes liste d above which is impacting customer churn so that appropriate business actions are taken b ased on these factors

```
In [7]: #print(tele.groupby('churn')['phone number'].count())
n_records = tele.shape[0]
n_churn= tele.groupby('churn')['phone number'].count()
#Since true is the second element taking secode element for percentage of churn ie n_churn[1]
greater_percent = (n_churn[1]/n_records)*100
print("The total number of customers",n_records)
print("The number of customers that didn't churn",n_churn[0])
print("The number of customers that churn",n_churn[1])
print("The percentage of customers churned",greater_percent)

The total number of customers 3333
The number of customers that didn't churn 2850
The number of customers that churn 483
The percentage of customers churned 14.491449144914492
```

The attributes:state,area code,phone number,international plan,voice mail plan are dropped as input attributes as they doesn't seem to be relevant to predict customer churn.

Count of NULL values is 0.

Count of Missing values is 0

```
In [115]: #Check Missing values
tele.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
state                     3333 non-null object
account length            3333 non-null int64
area code                 3333 non-null int64
phone number              3333 non-null object
international plan        3333 non-null object
voice mail plan           3333 non-null object
number vmail messages     3333 non-null int64
total day minutes         3333 non-null float64
total day calls           3333 non-null int64
total day charge          3333 non-null float64
total eve minutes         3333 non-null float64
total eve calls           3333 non-null int64
total eve charge          3333 non-null float64
total night minutes       3333 non-null float64
total night calls         3333 non-null int64
total night charge        3333 non-null float64
total intl minutes        3333 non-null float64
total intl calls          3333 non-null int64
total intl charge         3333 non-null float64
customer service calls    3333 non-null int64
churn                     3333 non-null bool
dtypes: bool(1), float64(8), int64(8), object(4)
```

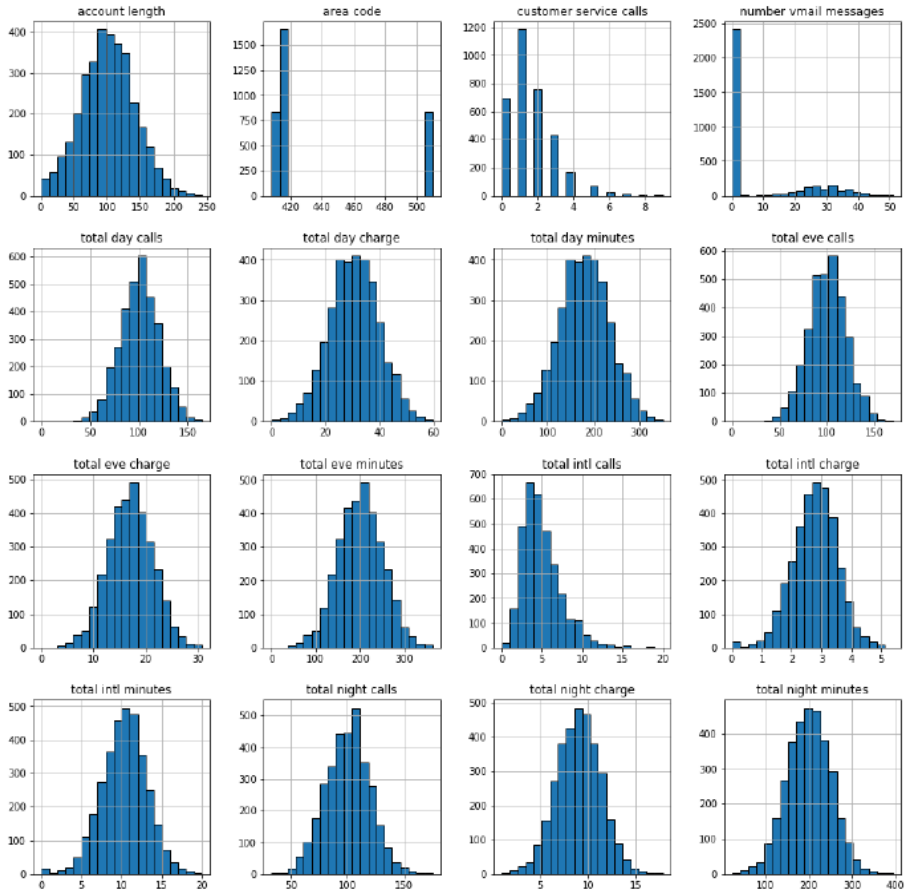
```
tele.isna().sum()
```

```
Out[114]: state                     0
          account length            0
          area code                 0
          phone number              0
          international plan        0
          voice mail plan           0
          number vmail messages     0
          total day minutes         0
          total day calls           0
          total day charge          0
          total eve minutes         0
          total eve calls           0
          total eve charge          0
          total night minutes       0
          total night calls         0
          total night charge        0
          total intl minutes        0
          total intl calls          0
          total intl charge         0
          customer service calls    0
          churn                     0
          dtype: int64
```

The data is split into 75-25 as train-test data out of the complete data

Exploratory Visualization

The following is the distribution of the input attributes



Based on the above picture looks like most of the features are normally distributed e.g

The following are the ranges of features:

1.Account Length 1-243

2.Total Day Calls 0-165

3.Total Day minutes 0-350

4.Total Day Charge 0-60

5.Total Night Calls 33-175

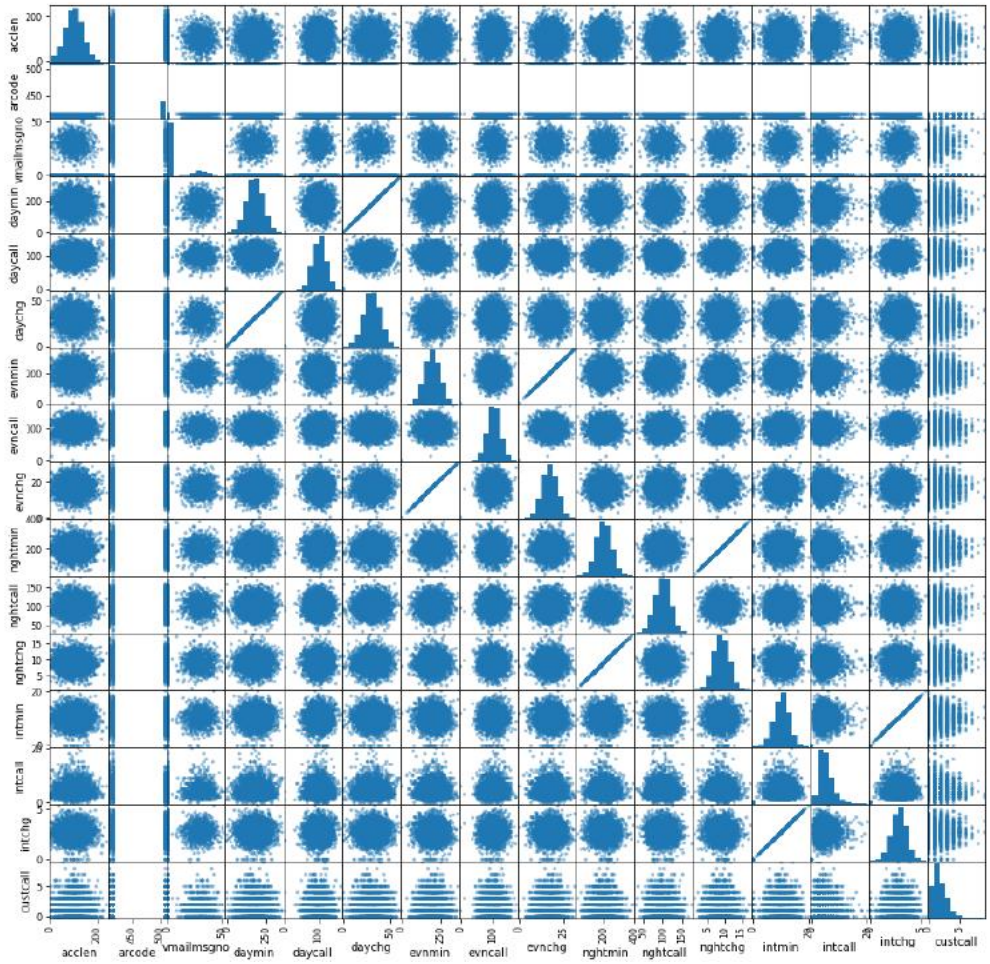
6.Total Night Charge 1-18

7.Total Night Minutes 23-395

Note area-code,customer service calls and number if vmail-messages doesn't seem to be normally distributed.

Also total internation calls are slightly skewed

The following is the scatter plot mix:



Also looking at the above scattered mix plot :

There seems to be a positive correlation between

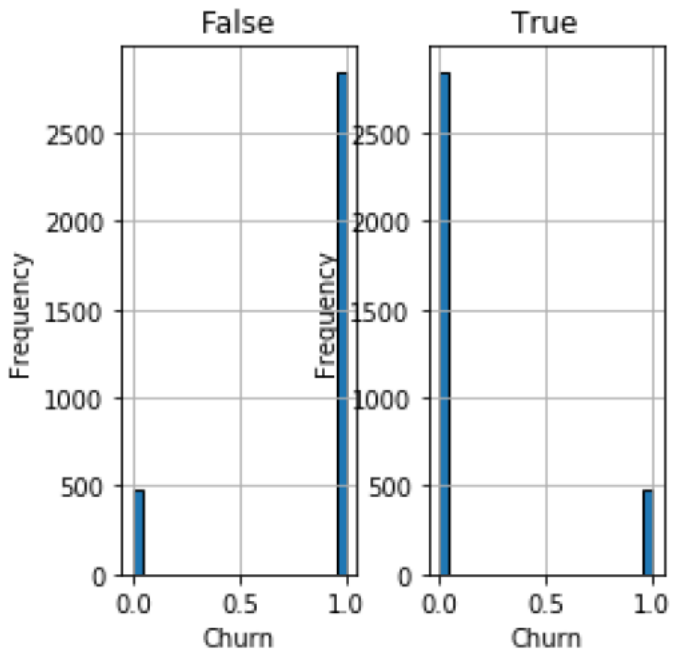
1.Day charge and Day minutes.

2.Evening Charge and Evening Minutes.

3.Night Charge and Night Minutes.

4.International Charge and International Minutes

Also the following is the output plot:



As mentioned earlier the churn percentage is about 15% based on the figure above

Data after feature engineering:

The attributes:state,area code,phone number,international plan,voice mail plan are dropped as input attributes as they doesn't seem to be relevant to predict customer churn

The no of input features after dropping the same would be: **15**

Algorithms and Techniques

The following supervised learning models currently available in `scikit-learn` that I choosed from:

- Decision Trees
- Ensemble Methods (Bagging, AdaBoost, Random Forest, Gradient Boosting)
- Logistic Regression

- **Linear Regression**

Linear regression is great when the relationship to between covariates and response variable is known to be linear (duh). This is good as it shifts focus from statistical modeling and to data analysis and preprocessing. It is great for learning to play with data without worrying about the intricate details of the model.

A clear disadvantage is that Linear Regression over simplifies many real world problems. More often than not, covariates and response variables don't exhibit a linear relationship. Hence fitting a regression line using OLS will give us a line with a high train RSS.

In summary, Linear Regression is great for learning about the data analysis process. However, it isn't recommended for most practical applications because it oversimplifies real world problems

- **Random Forest**
    - Good Real world example is Classification of Urban Remote Sensing Images [https://www.researchgate.net/publication/3204218_Decision_Fusion_for_the_Classification_of_Urban_Remote_Sensing_Images]
    - Strengths
        - can handle missing values
        - little to no tweaking required
        - can work in parallel
    - Weakness
        - bias for multiclass problems towards more recurring classes
        - difficulty in interpreting the model
    - Since, Random Forest are great on almost any machine learning problem, and will not get overfit like decision tree, I think it would be appropriate along with the fact that our dataset has categorical features.

Adaboost is a Boosting type Ensemble Learning Method. In the industry, boosting algorithms have been used for the binary classification problem of face detection where the algorithm has to identify wheter a portion of an image is a face or background (ref: https://en.wikipedia.org/wiki/Boosting_(machine_learning)). One of the main strenghts of Adaboost is that it is a fast algorithm, agnostic to the classifier and less prone to overfitting. During the iterative training, it continuously gives more weight to missclassified labels to allow the classifier to focus on the harder cases which increases the overall model's performance. On the other hand, noisy data and outliers in the data can negatively impact the performance so data pre processing is important. Furthermore, if a complex model is used as the base classifier, this can lead to overfitting to the training data. In my opinion, this model is a good candidate for the problem as our dataset is large yet clean. Therefore we will be able to perform multiple quick trainining iterations to maximize our overall accuracy on the unseen testing data

METHODOLOGY

Data Preprocessing

The attributes:state,area code,phone number,international plan,voice mail plan are dropped as input attributes as they doesn't seem to be relevant to predict customer churn.

Also the

Count of NULL values is 0.

Count of Missing values is 0

Since there are no null or missing values,nothing needs to be done for NULL values or missing values

All input variables are scaled between 0 and 1 using Scikit'sMin-Max scalar as part of data preprocessing

Implementation

To properly evaluate the performance of each model you've chosen, it's important that you create a training and predicting pipeline that allows you to quickly and effectively train models using various sizes of training data and perform predictions on the testing data. Your implementation here will be used in the following section. In the code block below, you will need to implement the following:

- Import `fbeta_score` from `sklearn.metrics`.
- Fit the learner training data and record the training time.
- Perform predictions on the test data `X_test`, and also train data set `X_train`.
    - Record the total prediction time.
- Calculate the F-score for both the training and testing set.
    - Make sure that you set the `beta` parameter!

BenchMark Model

We can used **Naïve predictor** as the benchmark model

Since we use use F-beta score as a metric that considers both precision and recall

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

In particular, when $\beta = 0.5$, more emphasis is placed on precision. This is called the F 0.5 score (or F-score for simplicity).

Looking at the distribution of classes(churn/not churn),its clear that most individuals doesn't churn.Hence its good to start with a statement that all customers would not churn. Making such a statement is **naïve** ,since no information is considered to substantiate this claim.It is always important to consider the naïve prediction for your data,to establish a benchmark for whether a model is performing well.

The following is the output after running three different models:

Results:

Metric	Linear Regression	Random Forest	Adaboost
F-train	0.2680	1.0000	0.5791
F-test	0.0684	0.7650	0.4749
Mean-score	0.08	0.7454	0.4000
Train-Time	0.0312	0.6772	0.2001
Prediction-Time	0.0	0.0534	0.0156

(Note the values of Benchmark Model:

```
Naive Predictor: [F-score: 0.1748]
```

)

Hence all classifiers f-score is better than the Benchmark Model

Looking at the results the Random Forest seems to outperform others though the train time is slightly higher compare to others.the value of F-train is 1 looks to be overfit but not we have better values even for F-test which means there doesn't seem to be overfit

Refinement:

The Model chosen out of the three i.e RandomForest is further refined using GridSearchCV.

The following parameters is chosen for model tuning

```
# TODO: Create the parameters list you wish to tune
parameters = {'n_estimators': [10,100,500,1000,2000],
              'criterion': ['gini', 'entropy']}
```

Where n_estimators is about no of trees in the forest and criterion is gini or entropy as its RandomForest Classifier.

With the tuned model,the f-score is as below:

Metric	Unoptimized Model	Optimized Model	BenchMark predictor
F-score	0.6896	0.7454	0.1748

Where BenchMark predictor is naïve-prediction and Unoptimized Model is for the RandomeForest (without grid search) and Optimized Model is with Grid Search

Regarding Challenges faced during the Journey:

Feature Scaling is a must when using regression.Initially,I trained models without scaling,and the results I got were nowhere near where they are now.Then after going through some articles on regression I learned how to scale data,and could able to do that

satisfactorily.

Dummy variables help avoid silly overwriting and save lot of time.I learned using dummy variables and used the same for output variable.I didn't had to used the same for input variables as all the non-numerical variables were dropped.

GridSearch tuning was something challenging and interesting.I learned how to used gridsearch and tune the model to perform better

Deriving feature importance too was something new to me and proving the same i.e getting f-score close to the actual with reduced features was also interesting

Robustness Check:

I deduced the feature importance and ran the model i.e RandomForest with the reduced set of features.I could see that with reduced set of features the Model's f-score is not reducing much as shown below:

```
Final Model trained on full data
------
The prediction on full data 0.7954545454545454
The prediction on reduced features 0.7193396226415094
```

Also if I go for cross validation

I don't see much reduction in F-score as shown below

```
Random Forest Classifier f train score = 1.0
Random Forest Classifier  f test score = 0.7650273224043715
Random Forest Classifier training time = 0.677161455154419
Random Forest Classifier  predict = 0.05343890190124512
```

```
rf_scores = cross_val_score(rf_clf, X, y, cv=5, scoring='f1')
print('Cross Validation scores using random forest \n',rf_scores)
print('Mean of Cross Validation scores',round(rf_scores.mean(),2))
results_rf['Mean_Score'] = round(rf_scores.mean(),2)
```

```
Cross Validation scores using random forest
 [0.64968153 0.66666667 0.68918919 0.64335664 0.73076923]
Mean of Cross Validation scores 0.68
```

The model looks to be robust based on the data above

Justification

Based on the data below:

Metric	Unoptimized Model	Optimized Model	BenchMark predictor
F-score	0.6896	0.7454	0.1748

And also with reduced features:

```
Final Model trained on full data
------
The prediction on full data 0.7954545454545454
The prediction on reduced features 0.7193396226415094
```

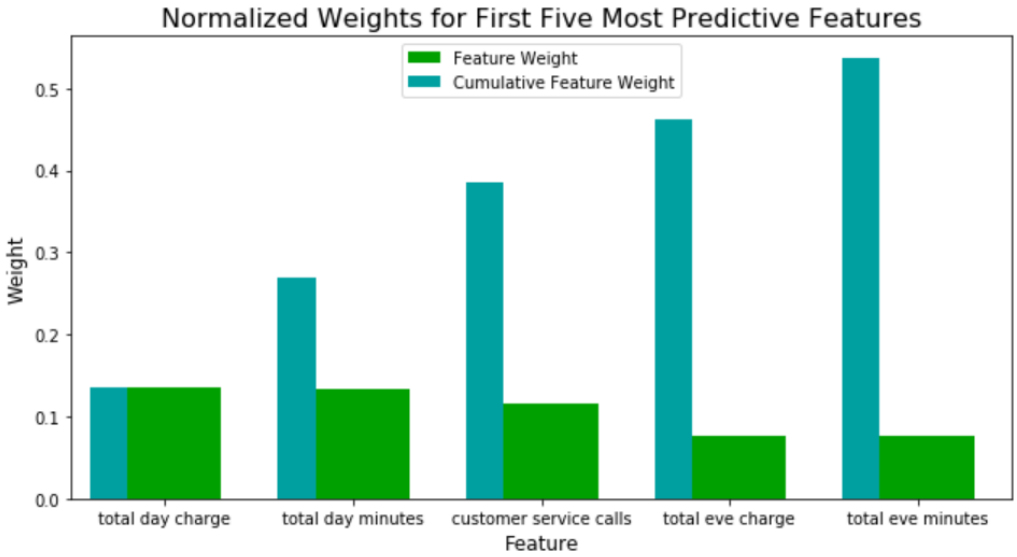
Looking at the Optimized results ie 0.7454/0.719(in case of reduced features) we can tell with some confidence that this model can be used by the Operator to predict Customer churn.

Since its not a mission critical task where we can expect F-score close to 1,we are still ok with 0.7454 and still go with this model for predicting customer churn.

CONCLUSION

Free form visualization

The five import features are deduced as below



Looking at the picture above total day charge and total day minutes seems to have have more feature weight compared to customer service calls or total evening charge/minutes

However the cumulative weightage of total evening charges/minutes is more compared to any other features.

Refection

The whole project experience can be summarized as follows:

The problem on which I didn't perform well during the coding challenge led me to explore it further.

Found the data set from IEEE URL as well on one of the Kaggle post.

Some base implementation was available but I wanted to improve on the same.

Visualize the data,did preprocessing by learning from other regression contests from Kaggle.

Preprocessing was an integral part of the project,using min-max scalar to scale the input attributes or finding out for missing/NAN values.

Applying selected algorithms and tuning the best performance model.

Using GridSearch Results which gave me better results.

It was difficult on figuring out to tune the Model.

Improvement:

Performing better feature Engineering.

Modifying parameters of GridSearch parameter space.n_estimators can have more set of values e.g 100,200,500,2000 etc

References

- https://www.kaggle.com/becksddf/churn-in-telecoms-dataset/data
- https://www.crowdanalytix.com/contests/why-customer-churn/
- https://github.com/ctufts/Cheat_Sheets/wiki/Classification-Model-Pros-and-Cons
- https://ieeexplore.ieee.org/document/8258400/
- https://ieeexplore.ieee.org/document/5705218/?reload=true