Machine Learning Engineer Nanodegree

Voice call quality prediction analysis

Sagar Chavan 21st Aug 2019

Contents

I. Definition

Project Overview

This problem is related to telecom domain. Mobile communication is growing the at rapid pace and there are multiple service providers in the market who provides services voice, internet, messages. Now mobiles are used beyond the just voice calls like mainly for internet usages video streaming. Industry is now entering the 5G (5^{th} generation) network services it is utmost important to pay attentions towards the quality of the services.

Overall 96% people are using mobiles phones worldwide and it is growing further [1]

When use telecom services is a need to today's world the quality of these services become main topic for both the parties that is for the customer as well as for service provider.

There are rules and regulations from TRAI related to quality of service located at TRAI website [2]

Data source is taken from Kaggle.

Problem Statement

Statement :-

Considering the growth in mobile usage and available service providers in the market there is a tough competition to catch the customer. Customer also looking for good quality services from the service provider. The quality of the services is different in different areas. It depends on many factors like service provider infrastructure, network type service provider providing, different service provider has different quality of service in the same area.

As part of this project we are going to predict the voice call quality based on given feature. This will help TRAI keep watch on service providers for their service quality. It can also help new customers which service provider is good in their area so they can choose based on it. It will help service providers as well to improve their service qualities.

Solution: -

From the problem statement we can clearly identify that this problem is classification problem where we must predict the voice call quality.

As part of solution we need to prepare/preprocess the data which should be best suited for our model. I will use the pandas and numpy lib for it. Then we have to design the model to predict the target label for that there are many available algorithms. I will mainly try decision tree, random forest and ensemble methods. Once models are ready we will evaluate the performance of the models based on

Outline of task: -

To solve the above stated problem, we need to follow the below steps as part of the solution.

1. I need to do the data exploration to see how many records are present in the data how many are falling in either categories (yes/no) and what is the percentage
2. I need to do data preparation and preprocessing like transform skewed features, normalize numerical features. We may need to do one hot encoding for non-numeric features.
3. Then I must split and shuffle the data for training and testing set.
4. I must transform latitude and longitude in to some other feature either to city/area
5. Build models as mentioned in the solution statement.
6. Train and test models
7. Then evaluate model based on F1 score.
8. Choose the best model based on the F1 score
9. Try to improve the F1 score by tuning the hyperparameters

Metrics

There are multiple evaluation metrics to evaluate the model performances like F1 score, ROC score. Here for this problem I am going to use the F1 score to evaluate the model performance.

Our data is imbalanced and F1-score performs well when the data is imbalanced while Accuracy and ROC score does not do well when data is imbalanced so I choose the F1-score metric

F1 score is calculated based on the precision and recall values using below formula.

F1score= 2 * (precision * recall / (precision + recall))

Here precision and recall are calculated based on the confusion matrix with the below formula.

Precision = TP/(TP+FP)

Recall = TP/(TP+FN).

II. Analysis

Data Exploration

This data is captured using customers feedback using TRAI MyCall App. Dataset having 2 files of data for the month of Apr2018 and May2018. Each record contains the below columns

1. Operator: - It is the telecom service provider name. for example Airtel, Rjio
2. Indoor_Outdoor_Travelling: - it is giving the position of the user when this call was made. Like when call was made whether person was travelling, or at home (Indoor) or outside.
3. Network type: - Network type means which technology is used to make a call whether it is 2G, 3G or 4G. for more detail read link
4. Rating: - It is the rating provided by the user for the call quality. Rating values are from 1 to 5. 5 is good quality
5. Voice call quality: - It is the category of the voice call quality. Whether user was satisfied with call quality. Or may call dropped, poor voice quality or poor network
6. Latitude: - It is the geographic point of earth on the world map
7. Longitude: - It is the geographic point of earth on the world map
8. State Name: - It is the state name from where user used service.

For some records network type can be unknown. latitude and longitude can be -1 or 0 that means user did not disclosed their location while providing the feedback.

Almost all columns are containing the categorical values, so we have done the one hot encoding of these columns. Latitude and Longitude columns are numeric but

instead of using those columns independently it is better to use it as single point (as location).

There are 95317 datapoints present in the dataset. Labels are structured like ‘Call dropped’, ‘Poor Voice Quality’ and ‘Satisfactory’, ‘Poor Network’.

Below is some more statistic related to each feature

Average Rating is 3.465499

operators and their count present in the dataset

Operator	count
RJio	39220
Airtel	22540
Vodafone	12510
BSNL	11612
Idea	8559
Tata	421
RComm	219
MTNL	144
Telenor	47
Other	45

user positions (indoor outdoor and Travelling) and their count

Indoor_Outdoor_Travelling	count
Indoor	63688
Outdoor	22423
Travelling	9206

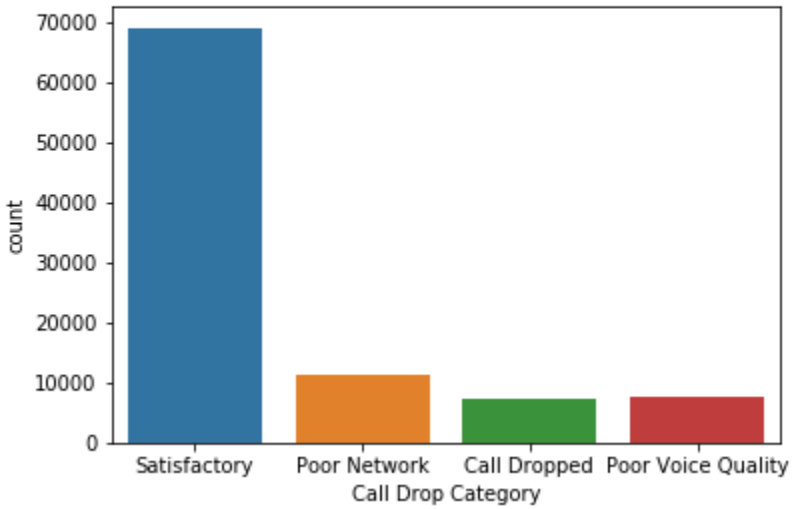
Network type and their count

Network Type	count
4G	52400
3G	19151
Unknown	18243
2G	5523

Exploratory Visualization

If we see the below bar graph it is clearly visible that data set is imbalanced. There are 4 different values for call category (target label) like ‘Call dropped’, ‘Poor Voice Quality’ and ‘Satisfactory’, ‘Poor Network’. In the below it is also shown count of each type of records. Call category records of ‘Satisfactory’ are much higher than other 3 categories. If we see logically like ‘Call dropped’, ‘Poor Voice Quality’ and ‘Poor Network’ are falls in the same category as ‘Poor quality’ and other ‘Satisfactory’ is ‘Good quality’ hence we will do binary classification instead of multiclass classification. By doing this we are balancing the dataset to some extent as well 68977:26340 that is approximately 2:1

```
Number of Satisfactory:  68977
Number of Poor NetWork :  11257
Number of Call Dropped:  7660
Number of Poor Voice Quality:  7423
```



Algorithms and Techniques

Our dataset is having latitude and longitude coordinates. We cannot drop those columns as it is very crucial data for our problem statement there are multiple ways to use latitude and longitude in algorithm but the easy technique to use it is by reverse geocoding. We are mapping all those latitude and longitude points to area using the reverse_geocoder lib.

There are many supervised learning algorithms available in the market like decision tree, Ensemble methods (Bagging, Adaboost, random forest), K-Nearest Neighbors and many others. To solve our problem, we have tried Decision Tree, Random forest and XGBoost algorithms. Based on the performance we choose the Random Forest algorithm to continue. To identify which features are important we just used feature_importances_ attribute of the random forest model.

To fine tune the chosen model, we used the GridSearch technique and make scorer module.

Problem I am solving as part of this project is classification problem falls under supervised learning category. So, Decision tree is the most basic algorithm for it.

Decision Tree Algorithm: -

Decision tree is like inverted tree which starts from most decisive feature at top then divides into subbranches. Then it picks next decisive feature and tree grows further. The Leaf nodes represents the final class/target label. The decision tree uses all the features splits it to create the branches.

In the decision tree all the nodes other than leaf are decision points.

Now how it is decided which feature is most decisive while creating the decision tree. How to create descending order of the important decisive features. If we change the order of the features in the decision tree it gives us different results. So, to get the correct prediction it is most important to create the decision tree with proper order of the feature.

To calculate which features are most important for decision tree it uses concept of entropy and information gain. Entropy is how much space is available for the particles to move around. Higher the space available then higher the entropy.

Entropy is calculated using the formula where p1 and p2 are the probabilities of classes.

$$entropy = -p1\log2(p1) - p2\log2(p2)$$

In case of multiple classes formula is

$$entropy = -p1\log2(p1) - p2\log2(p2) - ... - pn\log2(pn) = -i=1\sum npi\log2(pi)$$

Information gain is change in entropy. Which is calculated by formula

Information gain = Entropy(parent) – [m/(m+n)Entropy(child1 + n/(m+n) Entropy(Child2))]

Higher the information gains higher the importance of the feature. So once we get the features with higher information gain we can split he dataset based on that feature then continues similarly.

There are multiple hyperparameters exist for the decision tree to tune like max_depth , min_samples_split , min_samples_leaf

When there are many features in the dataset decision tree tends to overfit and it just memorize the data. Decision tree doesn't generalize well and small variation in the data might result into completely different tree.

Random Forest: -

Random forest is an ensemble model that create multiple decision trees with random bunch of features. It gives the prediction based on the average result from multiple decision trees in the forest.

It looks like the forest of decision trees, so it is called random forest.

Random forest reduces the chances of the overfitting.

There are multiple hyperparameters exist for the decision tree to tune like max_depth , min_samples_split , min_samples_leaf , max_features, n_estimators

XGBoost: -

It is boosted tree-based ensemble classifier like random forest. It automatically reduces the feature set, The accuracy of the XGboost is exceptionally good. But it takes higher time than random forest. It is most popular algorithm in data scientist.

Benchmark

For chosen problem we do not have any paper published or any other document which defines the benchmark model. For such classification problems decision tree is basic model. So, we can consider decision tree as benchmark model with default parameter. Whatever score we get from it is our benchmark model is the benchmark score

III. Methodology

Data Preprocessing

It is always required to use the clean, formatted and structured data for machine learning to get the better results. When we collect the data many times it is in the form of raw data. So, preprocessing is required for such data. As mentioned in the data exploration section our data set contains NA values for few records. It has invalid/missing values for some of the records.

As part of data preprocessing I performed below steps

1. Dropped the rows with NA values (mainly with 0 or -1 latitude and longitude).
2. Rating columns is directly related to call category which is our target variable, so we don't need that column hence Rating column is dropped
3. Did the reverse geocoding of the latitude and longitude columns and replaced State name column values with more accurate places (city or area name)
4. Then dropped latitude and longitude columns
5. Many columns having categorical values so did the one hot encoding. Before encoding dataset had around 8 columns after one hot encoding data set having 453 columns.
6. Mapped multiple classes to binary classification that is good quality and poor quality

Implementation

1. Libraries
   We are using libraries like pandas, numpy, seaborn, matplotlib, reverse_geocoder and sklearn

2. Split data: -

   First part in implementation is to divide the dataset into training and testing set as we do not have the separate dataset for testing. So, I divided the

dataset as 80:20 ratio. 80 % training and 20% in testing set. It is done using train_test_split technique of model_selection module.

3. Creating and Training models: -
   Created one method which can train model and predict target label then also calculate the accuracy score and f-score for the model
4. Initial model evaluation: -
   I choose 3 different algorithms to build the model those are decision tree, random forest and XGBoost all these 3 models will use the train_predict method defined in the above step and it calculates the performances of models for sample size 1%, 10% and 100% sample size.
5. Visualization of the performances: -
   Created barplot method using matplotlib module of the python to display the performances of the models in the form of bar graph. Bar graph plotted based on method created in previous step using the data calculated in step 3
6. Bar graph plotted based on method created in previous step using the data calculated in step 3
7. Chosen the best model based on model performances. For more details read Model Evaluation and validation section
8. Then tuned the selected model for multiple parameters. For more details read refinement section.
9. Challenges I faced: -
   The challenge I faced in the implementation is while converting the latitude and longitude to a city name/area name. There are multiple libraries present in the market (Geoplot and GeoPandas) and I tried couple of libraries few of them was not worked for me (maybe because I am new to this area).
   Then finally Reverse geocoder worked for me but when but when I was trying to call and replace the latitude and longitude in whole dataframe it was running very slow. It was working for small dataset but for my dataset it was taking lot of time (still it was not finishing so aborted the operation many times).
   As an alternative I loaded the dataset using csv module and reverse geocoded the dataset then appended to main dataframe loaded. And this approach worked!! It executes within a second.
   Second challenge I faced while model optimization. I used to lot of combination of hyperparameters but it is very time consuming process. It takes lot of time to execute the gridsearch block of my jupyter notebook.

After trying lot of combinations I finally kept only 3 values per parameter in the submitted jupyter notebook

Refinement

To improve the chosen model, I used gridsearch technique. I tuned the model for max depth n_estimators and min_samples_split parameters.

For max depth values I started tuning from max depth value 2, 4, 6 but ended up in 100, 150 and 180 values which gives better results than default model. I kept only last 3 values in the python code to keep it simple and easy to understand (reduce execution time as well). For n_eastimators I used 10, 20, 30 values and for min_samples_split I used 2, 4, 6 values

Finally got the accuracy score 0.7882 and F score 0.8241 for optimized model which is slightly better than the unoptimized model. It may get better score with more max depth, but it may tend to overfit the model so did not increased parameters further.

Final Result is as below

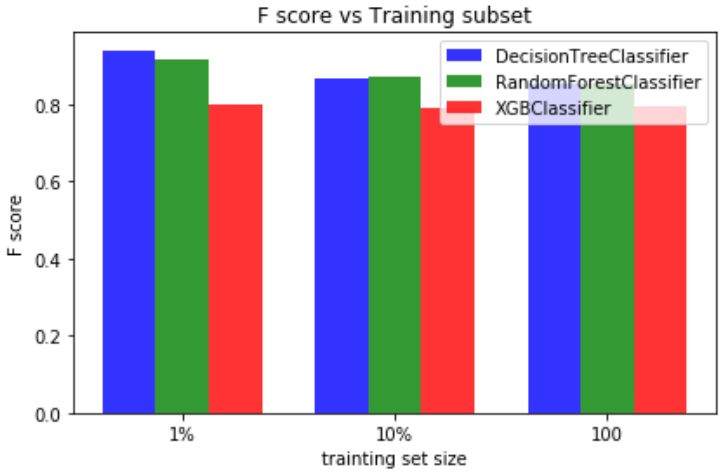
Metric	Unoptimized model	Optimized model
Accuracy score	0.7875	0.7882
F-score	0.8231	0.8241

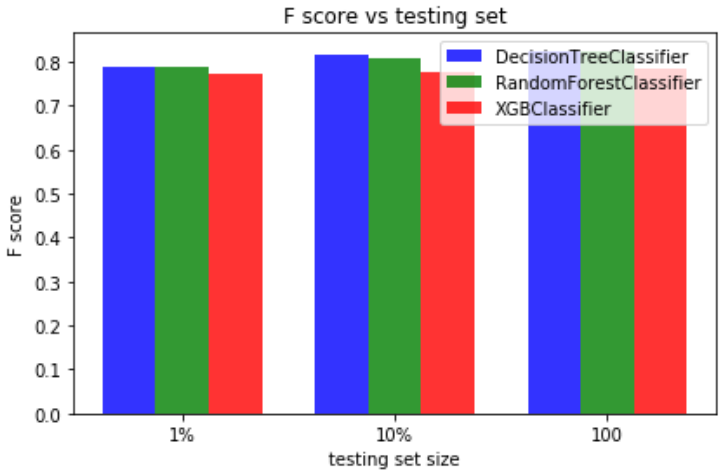
IV. Results

Model Evaluation and Validation

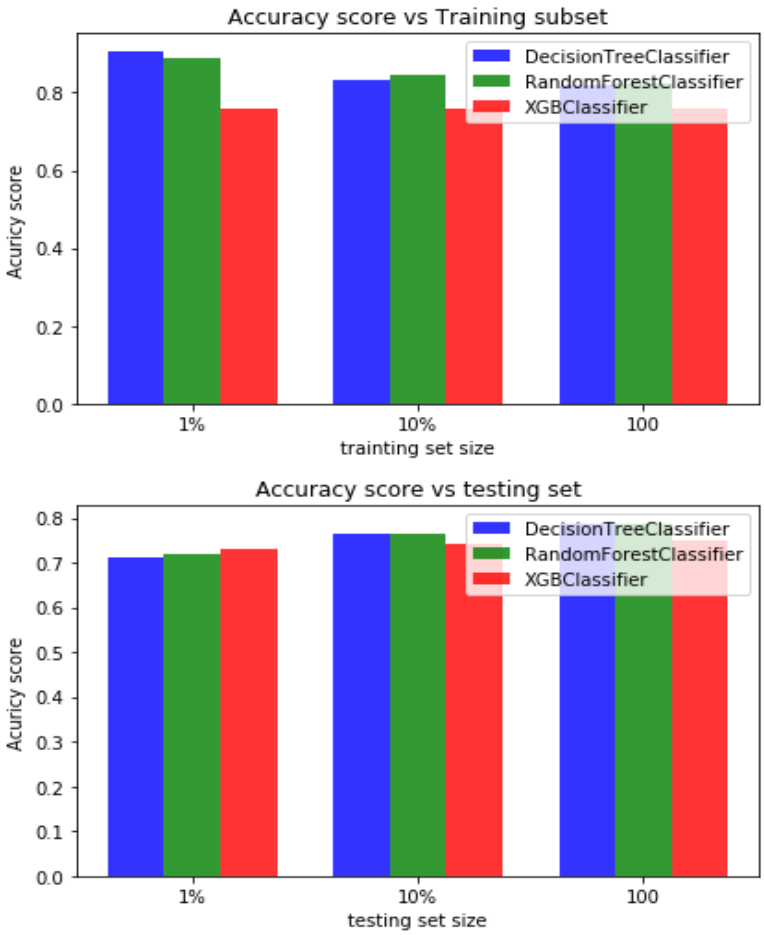
1. Choosing the best model: -
   Based on the performances of all 3 models I choose random forest model as best model as it gives better accuracy score and F-score for test set that is 0.787505 and 0.8231476. Decision tree model is also giving almost similar score, but decision tree model is tending to overfit easily. XGBoost is giving less score than random forest. Shown in pic 2 and 3

F-score graph:





Accuracy score:



After refinement of the model I got the final model as below

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=180, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=30,
n_jobs=None, oob_score=False, random_state=42, verbose=0,

warm_start=False)

Justification

Benchmark model itself gave us more than ~ 70% accuracy score and ~80% F-score which high. Then chosen final model after optimization gave me accuracy score 0.7882 and F-score 0.8241. Getting more than 80% F-score proves the final solution is good enough and giving the better prediction than just guessing it. When F score tends to 1 is considered as better model.

Metric	Unoptimized model	Optimized model
Accuracy score	0.7875	0.7882
F-score	0.8231	0.8241

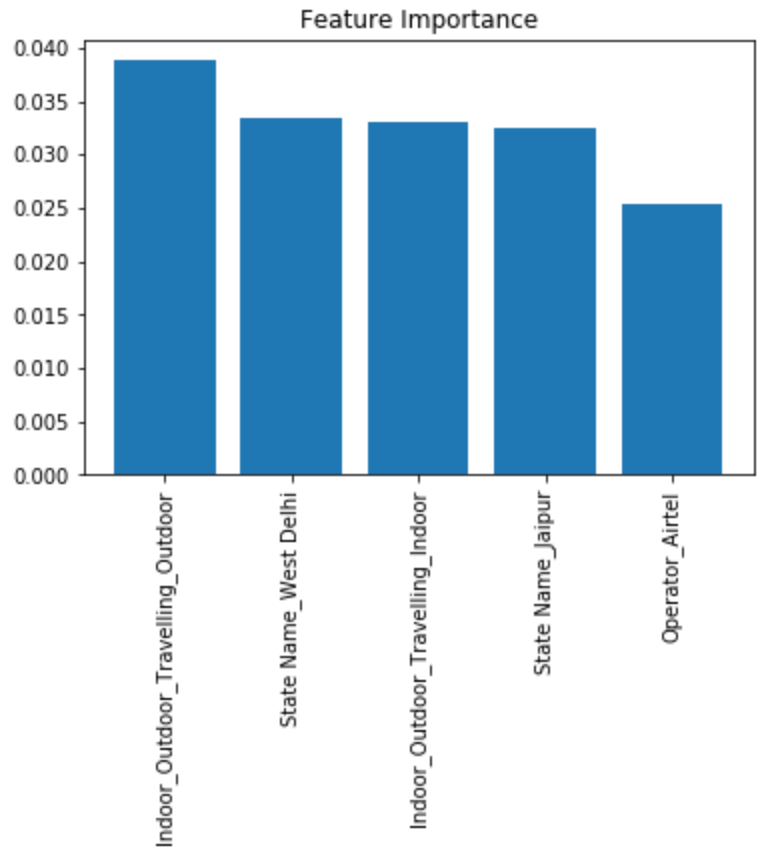
V. Conclusion

Free-Form Visualization

In our dataset there are only around 8 features present and all the columns having categorical values. I the initial look it looks less information but after the one hot encoding of the categorical values we get around 453 features.

In decision tree-based models it is very important how we choose the important features. Changing the features or order of features gives us completely different tree. More details mentioned in the algorithms and techniques section.

Out of which top 5 important features given by the random forest model is as below



From the feature importance graph, it is clearly visible that Indoor_outdoor_Travelling, State Name and Operator are the most important features. In the graph I showed only top 5 important features.

Reflection

To solve this problem I did dataset balancing, data exploration, preprocessing data, reverse geocoding of the latitude and longitude, build different model, train models chose best model and fine tune the selected model.

Wrote few helper methods as well for data visualization.

Basically, reverse geocoding and tuning the parameters was the difficult part in the entire process which took the considerable time in overall project time.

Improvement

There is still chance to improve this solution by tuning model again with more parameters.

In this solution I used reverse geocoding, but it is possible to build the model using small clusters (KNeighbors algorithm) instead of reverse geocoding. Then comparing the result of new model with existing model we may get better model.

There are may other algorithms those can be tried to build the model and based on performance comparison with this as a benchmark model we can improve the solution for this problem

References

[1] https://en.wikipedia.org/wiki/List_of_countries_by_number_of_mobile_phones_in_use

[2] https://main.trai.gov.in/telecom/qos

[3] https://www.kaggle.com/pranaysharma1108/real-time-voice-call-quality-data-from-customers

[4] https://pypi.org/project/reverse_geocoder/

[5]