CS5691: Pattern Recognition and Machine Learning

Shania Mitra	CH18B067
Abhishek Gopalan	CH18B032

February 6, 2021

---

1 Introduction to the datasets:

The following are the datasets provided:

1. tours.csv
2. tours_convoy.csv
3. bikers.csv
4. bikers_network.csv
5. train.csv
6. test.csv
7. locations.csv

1.1 tours.csv:

This dataset contains information about the location, organizer and description of the tour as can be seen in Figure 1. 'latitude' and 'longitude' were renamed to 'tour_latitude' and 'tour_longitude', while 'biker_id' was renamed to 'organizer_id'.

	tour_id	organizer_id	tour_date	city	state	pincode	country	tour_latitude	tour_longitude	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17	w18	w19	w20
0	VX4921758	DG47864012	31-10-2012	NaN	NaN	NaN	NaN	NaN	NaN	2	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	RT4999119	DE76440521	03-11-2012	NaN	NaN	NaN	NaN	NaN	NaN	2	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	SY28440935	FB7514445	05-11-2012	NaN	NaN	NaN	NaN	NaN	NaN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	RU82345152	HI1585781	30-10-2012	NaN	NaN	NaN	NaN	NaN	NaN	1	0	2	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
4	QP51165850	BA16098580	27-09-2012	NaN	NaN	NaN	NaN	NaN	NaN	1	1	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	1	2

Figure 1: Snippet of tours.csv; Shape of data: 3137972×110 ; Unseen columns are w21 - w100 and w_others

The column 'tour_date' contains many anomalous values such as 29th February in non-leap years, 31st on months with 30 days. Further dates from years from the future such as 2037 are also present. These are corrected by changing 29th February to 28th and correcting 31st to 30th while years beyond 2019 are clipped to 2019.

city	1557124
state	1889723
pincode	2686279
country	1533009
tour_latitude	1331880
tour_longitude	1331880

Figure 2: Columns with the presence of missing values along with the number of missing values

1.1.1 Extra features created

- 'tour_month', 'tour_day_of_week', 'tour_quarter': from tour_date [**Note:** 'tour_day_of_month', 'tour_year' were not created since dates were corrected to prevent usage of incorrect dates]
- 'imp_word_count' : sum of number of top 100 words $\left\{N_{imp} = \sum_{i=1}^{100} w_i\right\}$
- 'total_word_count' : sum of number of important words and other words $\left\{N_{total} = \left[\sum_{i=1}^{100} w_i\right] + w_{other}\right\}$
- 'imp_word_ratio' : ratio of number of important words and total number of words $\left\{\frac{N_{imp}}{N_{total}}\right\}$
- 'imp_word/w_other' : ratio of number of important words to number of other words $\left\{\frac{N_{imp}}{w_{other}}\right\}$
- 'tour_timezone' : timezone of location of tour in minutes given by 4 × tour_longitude

tour_day_of_week	tour_quarter	imp_word_count	total_word_count	imp_word_ratio	imp_word/w_other	tour_timezone
1	4	7	16	0.437500	0.777778	NaN
5	4	10	17	0.588235	1.428571	NaN
0	4	2	14	0.142857	0.166667	NaN
1	4	8	16	0.500000	1.000000	NaN
3	3	15	24	0.625000	1.666667	NaN

Figure 3: Extra features created in tours.csv

1.2 tours_convoy.csv

This dataset contains the details of bikers who were invited, going, not going etc. for a particular tour as can be seen in Figure 4.

	tour_id	going	maybe	invited	not_going
0	QQ59822043	BJ75964455 CF2302513 EC26086795 DI05886383 BE2...	CH33420590 FB7546982 BD50834692 FD2087573 FI31...	BH23091036 DH95873583 EB09144917 DF60622906 DB...	DF75574655 BA77296663
1	VX6467261	CD94228942 CG86116898 BA56558062 DH92942231 EB...	BE98184352 GE5689144 DH70076778 DD1335845 EC39...	BH88073374 HD3302094 BI30571649 GH6508092 HA81...	
2	QQ86208412		DD20380166 DI10793697	BD79121209 EE0668682	BH28988561 CJ50720854
3	RV21578336				
4	XU5842686	CE06118796 DF50897984 CJ4255260 BB25817077 BA9...	CG71721559 BH61448345 CD56975806 CG66669465 BA...	BF18670705 II0919237 CD26414227 CG73818347 DD2...	DF00235232

Figure 4: Guest list of bikers for tour; Shape of data: 24144 × 5

tour_id	0
going	1984
maybe	3169
invited	1822
not_going	6659

Figure 5: Number of missing values in tour_convoy

1.2.1 Extra features created:

1. 'num_going' / 'num_maybe' / 'num_not_going': Number of bikers going / maybe / not going
2. 'total_ num': number of bikers who responded $\{N_{total} = N_{maybe} + N_{going} + N_{not\,going}\}$ $_{(N:Number\ of\ people)}$
3. 'going_rate': Proportion of people going to total number $\left\{rate_{going} = \frac{N_{going}}{N_{total}}\right\}$ (similarly, 'not_going_rate')
4. 'acceptance_rate': Proportion of people who accepted the invitation out of those invited

$$\left\{acceptance = \frac{|S_{going} \cap S_{invited}|}{|S_{invited}|}\right\}_{(S:Set\ of\ people)}$$

	tour_id	num_maybe	num_going	num_invited	num_not_going	total_num	going_rate	not_going_rate	acceptance_rate
0	QQ59822043	7	7	70	2	16	0.437500	0.125000	0.0
1	VX6467261	8	11	75	1	20	0.550000	0.050000	0.0
2	QQ86208412	2	1	2	2	5	0.200000	0.400000	0.0
3	RV21578336	1	1	1	1	3	0.333333	0.333333	0.0
4	XU5842686	6	6	10	1	13	0.461538	0.076923	0.0

Figure 6: Values of new features created corresponding to each 'tour_id' in tour_convoy.csv; Added 8 new features

1.3 bikers.csv

This dataset contains the information of biker location, language spoken, timezone, etc. Using locations.csv we can also get the biker's latitude and longitude information. 'latitude' and 'longitude' were renamed to 'biker_latitude' and 'biker_longitude' while 'time_zone' was renamed to 'biker_timezone'.

biker_id	0
language_id	0
location_id	0
bornIn	0
gender	109
member_since	57
area	5464
biker_timezone	436

Figure 7: Number of missing values in bikers.csv

	biker_id	language_id	location_id	bornIn	gender	member_since	area	biker_timezone	biker_latitude	biker_longitude
0	DB97468391	id	ID	1993	male	02-10-2012	Medan Indonesia	480.0	3.589665	98.673826
1	DF37982273	id	ID	1992	male	29-09-2012	Medan Indonesia	420.0	3.589665	98.673826
2	IC3183725	en	US	1975	male	06-10-2012	Stratford Ontario	-240.0	43.370090	-80.981802
3	BI72223848	en	US	1991	female	04-11-2012	Tehran Iran	210.0	35.689252	51.389600
4	DE29017717	id	ID	1995	female	10-09-2012	NaN	420.0	NaN	NaN

Figure 8: Snippet of bikers.csv along with bikers' latitude and longitude information from locations.csv; Shape of data: 38209×10

The column 'bornIn' which is supposed to contain the year of birth of the biker contains some anomalous values such as '23-May','−−None'. These are replaced by the mode year of birth: 1993.

1.4 bikers_network.csv

This dataset contains the list of 'biker_ids of friends of each biker as can be seen in Figure 9.
The 'friends' column has 139 missing values. Such bikers are assumed to have no friends.

	biker_id	friends
0	DB97468391	BD46449342 DI73244116 EC26080662 BC22907620 FE...
1	DF37982273	BE91560444 DJ5798035 CA36380346 IJ9375619 DF34...
2	IC3183725	BE84954627 BJ50387873 BG52977611 EB85960823 EC...
3	BI72223848	ID361640 HC3814682 FF7944478 BH24049724 CF3059...
4	DE29017717	EC53303705 CB30310957 BI38389374 DJ28735761 HB...

Figure 9: List of biker friends of each biker in bikers_network.csv; Shape of data: 38202×2

1.4.1 Extra features created:

'num_friends': number of friends of each biker obtained from the 'friends' column

---

2 Imputation of missing values:

Two copies of the data were created. On one, imputation was performed and while the other was left unimputed. This is because most modern day models such as LightGBM and XGBoost can handle missing values. Thus, we feed both the datasets to the models to see which performs better.

2.1 bikers.csv:

1. 'gender': Filled in according to the probability distribution of genders [For example, in bikers, among the entries that do not have gender missing, 62% are male and 38% are female. Thus, to fill in the missing values for rows which do not have gender, we pick the value 'male' with 62% probability and the value 'female' with 38%]

2. 'member_since': filled according to the condition probability distribution of 'member_since' given year of birth ('bornIn') [The dataset is grouped by 'bornIn'. In each such group, imputation of 'member_since' is performed by looking at the probability distribution of non-empty rows of 'member_since' of that particular group.]

3. 'biker_timezone': filled according to the conditonal distribution of timezones given area

2.2 tours.csv

2.2.1 Location of tour:

- First, we look for 'organizer_id' in bikers.csv to see if the location of the organizer is available. The assumption is that the organizer will organize the tour in his/her own city.
- If not available, we look for 'friends' of the biker in 'bikers_network' and find the mode of location of the organizer's friends. We assume that the organizer has maximum friends from his/her own city.
- If this is also not available, we look for 'tour_id' in 'tour_convoy' and take the mode of location of those that are going. The assumption is that maximum number of people going are from the same place as the tour, since people are more willing to go to tours in their own city. Therefore tour location is assumed to be the mode of locations of people going for that particular tour.
- However, even after these steps, some missing values remain. These are left to be treated upon merging with the train and test data.

2.2.2 Tour latitude and longitude

Missing values of 'tour_latitude' and 'tour_longitude' were obtained from feeding location to geopy geocoder.

3 Preparing train and test data:

	biker_id	tour_id	invited	timestamp	like	dislike
0	DA44012	QY18771225	0	02-10-2012 15:53:05	0	0
1	DA44012	QU02284248	0	02-10-2012 15:53:05	0	0
2	DA44012	RU29072432	0	02-10-2012 15:53:05	1	0
3	DA44012	SP72478280	0	02-10-2012 15:53:05	0	0
4	DA44012	QS90707377	0	02-10-2012 15:53:05	0	0

Figure 10: Snippet of train.csv provided; task is to predict 'like'; merging with other datasets takes place on 'biker_id' and 'tour_id'

'train' and 'test' were merged with 'bikers.csv' on 'biker_id' and with 'tours.csv' on 'tour_id' while extra features from tour_convoy and bikers_network were also merged based on 'tour_id' and 'biker_id' respectively. Two such merges were performed - one with all unimputed datasets and one with all imputed datasets. As explained above, this was done to check on which one the models would perform better. After merging, 'timestamp' is renamed to 'biker_inform_date'.

3.1 Extra features added in final training and testing data:

Earlier we were only able to add features to datasets derived from their own features. Once all datasets are merged, it was possible to create features combining columns from multiple datasets.

1. 'month_inform', 'day_inform', 'quarter_inform': features from timestamp at which biker was informed about the tour

2. 'inform_month_is_tour_month': whether the biker came to know about the tour in the same month as it is going to be held (similarly 'inform_quarter_is_tour_quarter')

3. 'member_tour_period': number of days between the biker becoming a member and the tour date

4. 'inform_tour_period': number of days between the biker finding out about the tour and the tour date

5. 'timezone_diff': difference between timezone of tour and biker

6. 'distance_between_biker_tour': distance in km between biker and tour location using their respective latitudes and longitudes

7.a 'num_friends_going': number of friends of the biker who have confirmed they are going for the tour (from 'bikers_network' and 'tour_convoy')

7.b 'num_friends_invited', 'num_friends_maybe', 'num_friends_not_going': similar to 'num_friends_going'

8. 'is_weekend': whether the tour is on a Saturday/Sunday or not.

9. 'lat_diff': difference in latitudes of the tour and biker [**Note:** difference between longitudes was not considered because it corresponds to the difference in timezones, which is already accounted for)

10. 'age': age of biker during tour ('tour_date' − 'bornIn')

11. 'friends_going_ratio': number of friends going for the tour divided by total number of people going

12. 'friends_with_organizer': whether the biker is friends with the organizer or not

3.2 Imputation of missing values

For the rows that still remained undetermined in the imputed copy , missing values in 'biker_longitude' and 'biker_latitude' were filled according to the conditional distribution of latitude/longitude given 'location_id', while those in 'tour_longitude' and 'tour_latitude' were filled with the medians of corresponding columns of training data. The unimputed copy was left as it is. While feeding the data to the model, the following were dropped:

- **Dates:** 'tour_date', 'biker_inform_date','member_since','bornIn'
- **Areas:** 'city', 'state','pincode','country', 'area'
- **IDs:** 'biker_id', 'tour_id', 'organizer_id'

Final number of features: 145
Number of training samples: 13866
Number of test samples: 2690

---

4 Model Details:

4.1 Feature Extraction

Recursive feature elimination with cross-validation (RFECV) was used to determine the number and features to give best model performance. The underlying estimator used was random forest. The model selected 98 of the 145 features. However, in the final model, all features together gave better performance than the selected subset and hence all features were used. Feature importances obtained from the final model are in Figure 11.

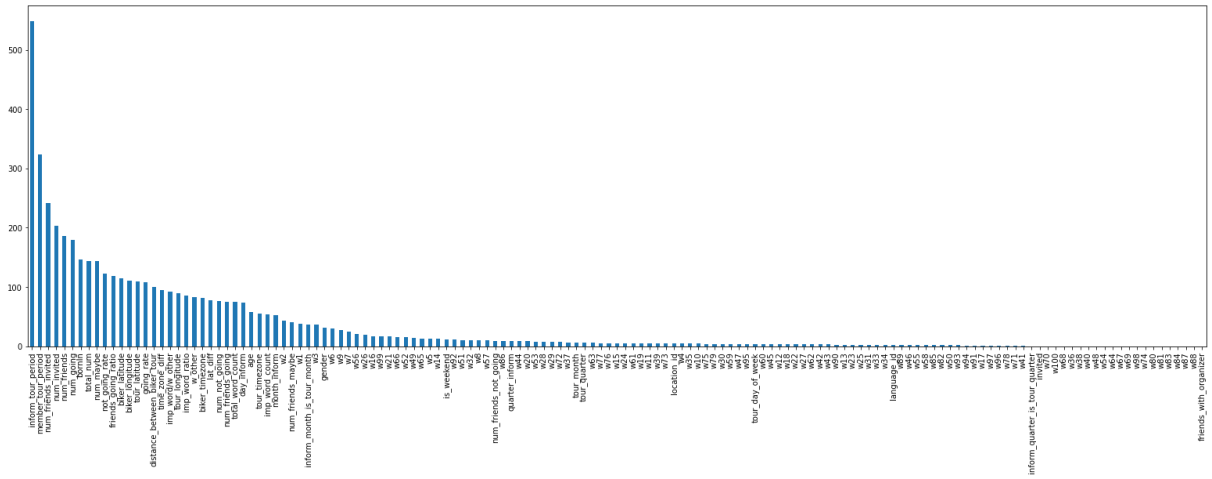

Figure 11: Feature Importances of 145 features used; We can see that new features created are far more important than the existing ones

As we can see, 'inform_tour_period' and 'member_tour_period' are the most informative features, while most of the w_i are not very informative. Further, we can see that continuous features are more informative than the categorical features.

4.2 Gradient Boosting Trees

To perform predictions the following tree models were trained and evaluated using 20% validation data.

1. XGBoost
2. Catboost
3. LightGBM

Using these models and both - the imputed and unimputed copies of data, we ran 3-fold cross-validation using Random Search to obtain the hyperparameters. LightGBM outperformed the other models in this initial search with a 3-fold cross-validation AUC of 0.778 and logloss of 0.457. Hence, the final model used is LightGBM with unimputed data. After this, detailed hyperparameter search was done on this model.

4.3 Hyperparameter Tuning

The following 7 hyperparameters were tuned:

num_leaves	n_estimators	learning_rate	max_depth
bagging_fraction	feature_fraction	min_data_in_leaf	

Various methods were used to obtain the best set of hyperparameters. These include:

4.3.1 Random Search:

Used sklearn's function RandomizedSearchCV for random search of hyperparameters, using 3-fold cross-validation and ROC-AUC scoring. Maximum ROC-AUC obtained = 0.778
Validation Accuracy for this model = 76.5%

4.3.2 Bayesian Optimization:

Used the Bayesian Optimization library to tune hyperparameters and performed 400 iterations.
Maximim ROC-AUC obtained = 0.7656
Validation accuracy of best model = 77.4%

4.3.3 Tree Parzen Estimator:

Used Optuna to tune hyperparameters, performing 1200 iterations.
Maximum ROC-AUC obtained = 0.7834
Maximum Validation Accuracy = 80.1%
Thus, hyperparameters obtained using TPE give the best performance.

4.4 Details of LGBM:

Task: The training set provides information on whether a biker likes a particular tour or not. Thus we translated the ranking problem into a classification task where we use LGBM to predict whether the biker likes the tour or not. To output the final ranks of tours, we arranged the tours corresponding to a particular biker in decreasing order of like probabilities predicted by the model.

Hyperparameter tuning using Optuna: The hyperparameter tuning library Optuna is used to obtain the values of optimal hyperparameters. In this, the search space of various hyperparameters to be tuned are specified. A study with the objective function is defined and trials run in which various values of hyperparameters are picked from the search space and its performance is noted (based on the objective). As the trials proceed the search space is narrowed down to explore regions around values that have worked well earlier. In our case the objective function was taken to be the mean-AUC on 3-fold cross-validation on the training set, which was to be maximized. A total of 1200 trials were run.

4.4.1 Optimal regions for hyperparameters

Figure 12: Plot of Objective Value vs Hyperparameter Choice

In this plot we can observe the how various regions of hyperparameters perform. On the x-axis of each plot we have the range of the hyperparameter specified in the search space, while on the y-axis we have the objective value. The shades of blue represent the trial at which that particular value is used. Darker the shade, more recent the trial. We can see that in the later trials of the study, only values that are known to perform well are chosen. We can also see that higher values of bagging fraction perform tend to perform better, even though there are some exceptions. We notice that the objective value does not fall much even for sub-optimal values of bagging fraction. This is also true for feature fraction whose optimal value is seen to lie between 0.5 and 0.6 . In case of learning rate we can clearly see that performance drops sharply as learning rate moves away from the optimal region whereas in case of max_depth, n_estimators, num_leaves and min_data_in_leaf, performance remains more or less constant across all values. Thus, learning rate and bagging fraction are the key decision makers of performance.

4.4.2 Picking combinations of hyperparameters

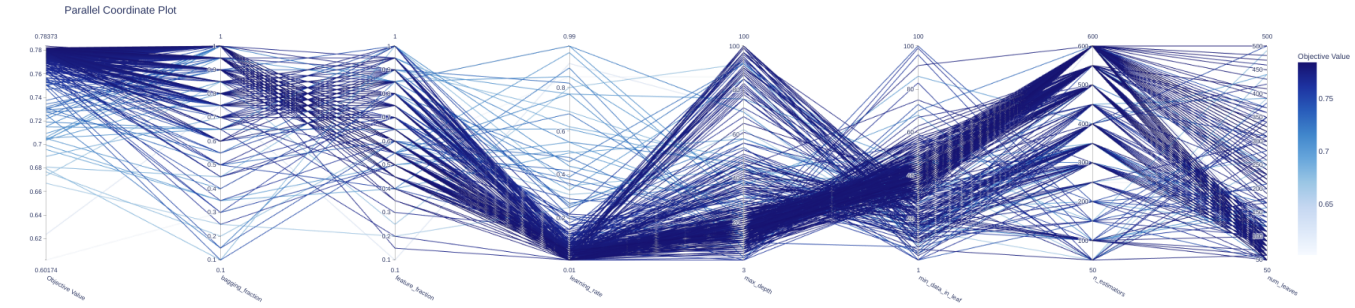

Figure 13: Plot of combinations of Hyperparameters chosen during trials; we can see that combinations that performed better were picked more often.

This plot shows us the various combinations of hyperparameters tried and the colour of the line tells us the ROC-AUC value. We can see that in the values of hyperparameters that give higher ROC-AUC values have been picked more in the trials. This is done because in the hyperparameter space, values close to eachother give similar objective values.

5 Performance of Models

Sl. No	Bagging Fraction	Feature Fraction	LR	Max Depth	min data in leaf	Num estimators	Num leaves	Val AUC	Val logloss	Public board Score	Private uboard Score
1	0.6	0.7	0.01	80	15	200	100	0.77560	0.4568	0.76780	0.69858
2	1	0.5	0.04	52	16	500	113	0.7848	0.4506	0.76490	0.71020
3	**0.95**	**0.6**	**0.025**	**29**	**37**	**500**	**116**	**0.78277**	**0.4516**	**0.76277**	**0.71978**
4	1	0.7	0.03	14	29	550	100	0.78299	0.4515	0.76030	0.71172

Models submitted: 1,2

Best performing model: 3

The hyperparameters of model-1 are obtained using random search, while hyperparameters of models 2, 3 & 4 have been tuned using Optuna. All the models listed here are LightGBM models trained on unimputed data. We can see that neither AUC nor logloss correlate perfectly with the final evaluation metric. Futher the models trained by Optuna outperform those tuned by random search. Models - 2,3,4 are close to eachother in terms of hyperparameter choices as compared to model-1. From this we can conclude that the optimal range for Learning Rate (LR) $\approx 0.03 - 0.04$ and Number of estimators ≈ 500 for this dataset.

---

6 Summary

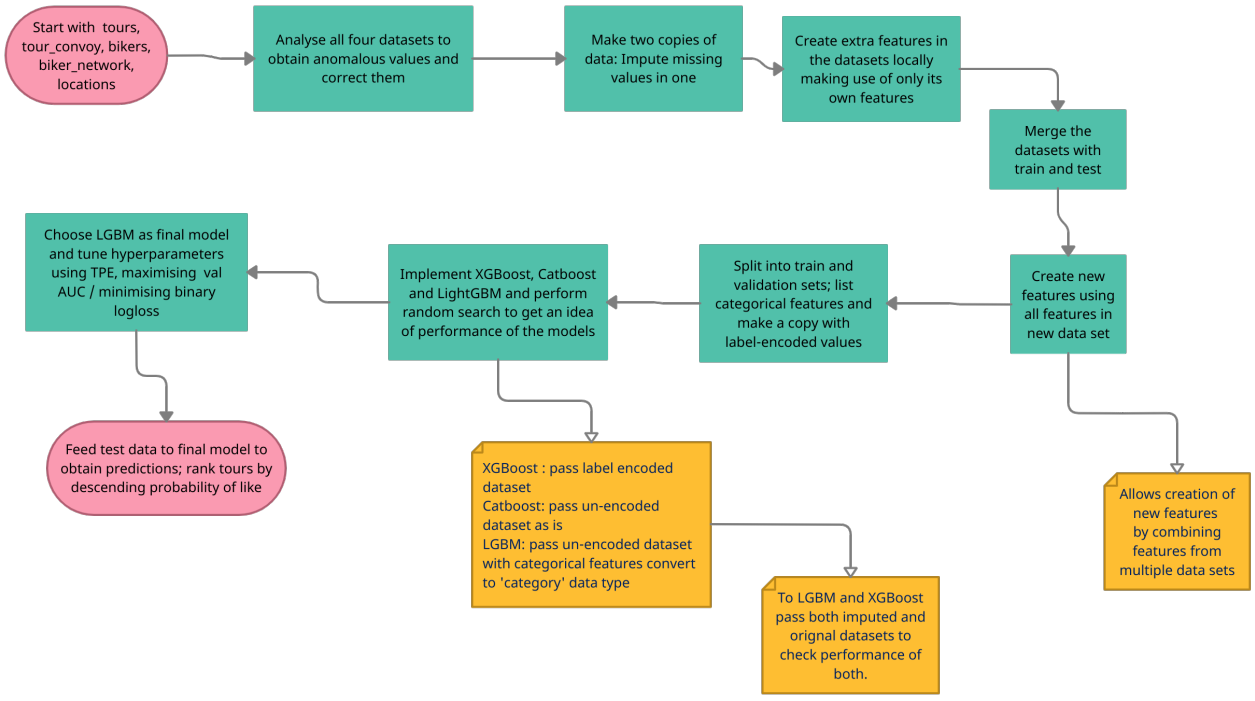

Figure 14: Flowchart describing procedure adopted
